Trabajo Final
Especialidad en Ingeniería de Sistemas Expertos

HERRAMIENTAS INTELIGENTES PARA EXPLOTACIÓN DE INFORMACIÓN

Autora: Ing. María Alejandra Ochoa

Directores: Dr. Ramón García Martínez
M. Ing. Paola Britos

2004

ÍNDICE

Contenido	Página
1. Introducción	**3**
2. Fundamentos	**5**
2.1. Backpropagation	**5**
2.2. SOM (Self Organizing Map)	**6**
2.3. C4.5	**7**
3. Aplicación	**8**
3.1. Backpropagation	**8**
3.2. SOM (Self Organizing Map)	**12**
3.3. C4.5	**14**
4. Herramientas para explotación de datos	**24**
4.1. Herramienta de aplicación de Backpropagation (NNclass)	**24**
4.1.1. Funcionamiento de la herramienta	**24**
Hoja ReadMe	**24**
Hoja User Input	**27**
Hoja Data	**28**
Hoja Calc	**29**
Hoja Output	**30**
Hoja Profile	**31**
Hoja LiftChart	**32**
4.1.2. Bases de datos de ejemplo	**33**
4.2. Herramienta de aplicación de SOM (NNclust)	**34**
4.2.1. Funcionamiento de la herramienta	**34**
Hoja ReadMe	**34**
Hoja Input	**35**
Hoja Data	**36**
Hoja Weights	**36**
Hoja Output	**37**
Hoja Junk	**37**
Hoja Plot	**37**
4.2.2. Bases de datos de ejemplo	**38**
4.3. Herramienta de aplicación de C4.5 (CTree)	**39**
4.3.1. Funcionamiento de la herramienta	**39**
Hoja ReadMe	**39**
Hoja User Input	**41**
Hoja Data	**42**
Hoja Calc	**43**
Hoja Output	**44**
Hoja Profile	**45**
Hoja LiftChart	**46**
4.3.2. Bases de datos de prueba	**47**
5. Conclusión	**48**
6. Bibliografía	**49**

1- INTRODUCCIÓN

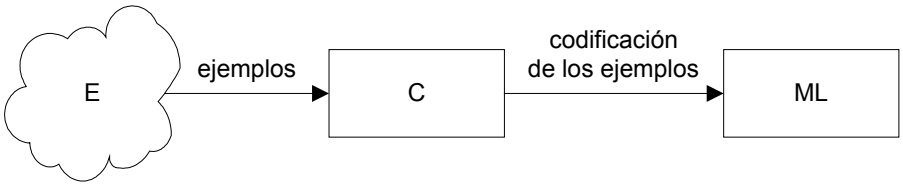
El ***Aprendizaje Automático*** (Machine Learning): se enfrenta con el desafío de construir programas computacionales que automáticamente mejoren con la experiencia. Estos son sistemas capaces de adquirir conocimientos de alto nivel y/o estrategias para la resolución de problemas mediante ejemplos, en forma análoga a la mente humana. A partir de ejemplos provistos por un tutor o instructor y de los conocimientos de base o conocimientos previos, el sistema de aprendizaje crea descripciones generales de conceptos.

Minería de Datos (Data Mining): busca generar información similar a la que podría generar un experto humano, que además satisfaga el principio de comprensibilidad. El objetivo de éste es descubrir conocimientos interesantes; como patrones, asociaciones, cambios, anomalías y estructuras significativas a partir de grandes cantidades de datos almacenados en bases de datos, data warehouses, o cualquier otro medio de almacenamiento de información.

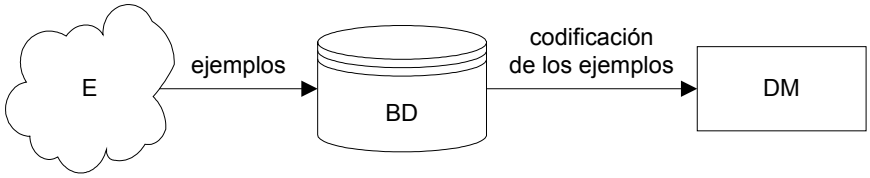
Redes Neuronales Artificiales (RNA): Son modelos que intentan reproducir el comportamiento del cerebro. Del mismo modo que aquel, realiza una simplificación, averiguando cuáles son los elementos relevantes del sistema. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea. Este modelo posee dispositivos elementales de proceso, las neuronas, al igual que en el modelo biológico. El conjunto de estas pueden generar representaciones específicas, por ejemplo un número, una letra o cualquier otro objeto.

Algoritmos de Inducción: Algunos utilizan ejemplos como entradas para aplicar sobre ellos un proceso inductivo y así presentar la generalización de los mismos como resultado de salida. Existen dos tipo de ejemplos, los positivos y los negativos. Los primeros fuerzan la generalización, mientras que los segundos previenen que ésta sea excesiva. Se pretende que el conocimiento adquirido cubra todos los ejemplos positivos y ningún ejemplo negativo. Los ejemplos deben ser representativos de los conceptos que se está tratando de enseñar. Además la distribución de las clases en el conjunto de ejemplos de entrenamiento, a partir de los que el sistema aprende, debe ser similar a la distribución existente en los datos sobre los cuales se aplicará el sistema. Otros utilizan el descubrimiento y observación, para que el sistema forme criterios de clasificación. En este caso se aplica aprendizaje no supervisado, permitiendo que el sistema clasifique la información de entrada para formar conceptos. En estos casos el sistema puede interactuar con el entorno para realizar cambios en el mismo y luego observar los resultados.

La ***Minería de Datos*** es un caso especial del ***Aprendizaje Automático***, utiliza sus métodos para encontrar patrones, con la diferencia que el escenario observado es una base de datos. En un esquema de Aprendizaje Automático, el mundo real es el entorno sobre el cual se realiza el aprendizaje, estos se traducen en un conjunto finito de observaciones u objetos que son codificados en algún formato legible. El conjunto de ejemplos constituye la información necesaria para el entrenamiento del sistema.



En un sistema de minería de datos, la codificación es reemplazada por una base de datos que refleja algún estado del entorno.



Los sistemas de aprendizaje se clasifican en métodos de caja negra y métodos orientados al conocimiento. Los primeros desarrollan su propia representación de conceptos, realizando cálculos numéricos de coeficientes, distancia, vectores; generalmente no comprensibles por los usuarios. Los segundos tratan de crear estructuras simbólicas de conocimiento que si sean comprensibles para el usuario. Las ***Redes Neuronales*** pertenecen al primer grupo y el ***Aprendizaje Automático*** al segundo. [García Martínez et al, 2003]

2- FUNDAMENTOS

2.1 Backpropagation

En 1986 Rumelhart, Hinton y Williams, desarrollaron un método que tenía por objetivo lograr que una red neuronal aprendiera la asociación que existe entre los patrones de entrada que ingresan a la misma y las clases correspondientes, utilizando más niveles de neuronas de los que empleó Rosenblatt en su Perceptrón. El método está basado en la generalización de la regla delta y logra ampliar considerablemente el rango de aplicación de las redes neuronales.

Backpropagation o algoritmo de propagación hacia atrás o retropropagación es una regla de aprendizaje que se puede aplicar en redes con más de dos capas de neuronas. Una de sus características es la capacidad de representar el conocimiento en las capas ocultas, logrando así cualquier correspondencia entre entradas y salidas.

El funcionamiento de una red backpropagation consiste en el aprendizaje de un conjunto predefinido de pares entrada-salida dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases. Primero se aplica un patrón de entrada en la capa de entrada de la red, que luego se propaga hacia las capas superiores hasta generar una salida, se compara el resultado obtenido en cada neurona de salida con el valor deseado para esa neurona y se obtiene un error para dicha unidad. A continuación, estos errores se transmiten hacia atrás, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa hasta llegar a la entrada y hasta que cada neurona haya recibido un error que describa su aporte al error total. Según el valor del error recibido, se reajustan los pesos de las conexiones entre cada par de neuronas en la red, de manera de que el error total cometido para ese patrón disminuya. [García Martínez et al, 2003]

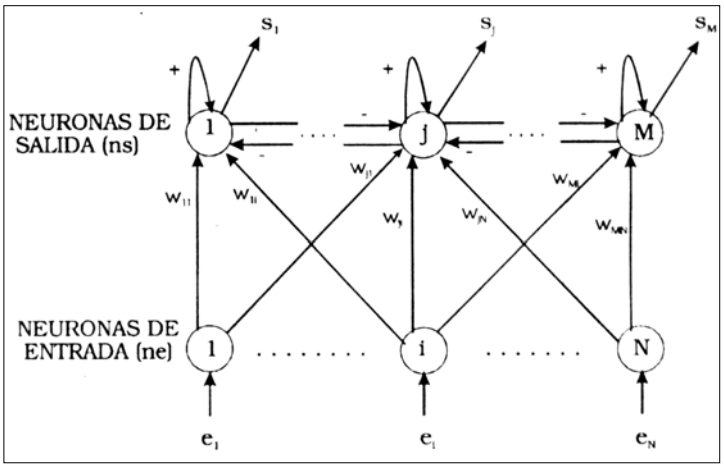
2.2 SOM

En 1982, Teuvo Kohonen, presentó un modelo de red neuronal, basado en el funcionamiento de neuronas biológicas. La red neuronal diseñada posee la capacidad de formar mapas de características. El objetivo de Kohonen era demostrar que un estímulo externo por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas.

El modelo tiene dos variantes, LVQ (Learning Vector Quantization) y TPM (Topología Preserving Map) o SOM (Self Organizing Map). Ambas se basan en el principio de formación de mapas topológicos para establecer características comunes entre las informaciones (vectores) de entrada a la red, aunque difieren en las dimensiones de éstos, siendo de una sola dimensión en el caso de LVQ y bidimensional e incluso tridimensional en la red SOM o TPM

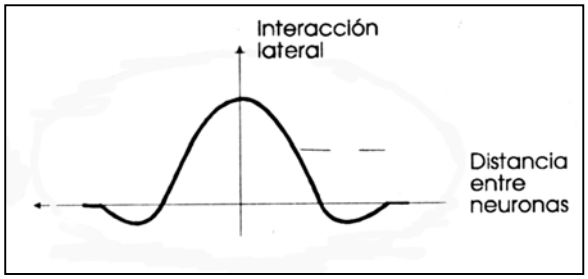
El modelo presenta dos capas con N neuronas de entrada y M de salida. Cada una de las N neuronas de entrada se conecta a las M de salida a través de conexiones hacia adelante (feedforward).

Entre las neuronas de la capa de salida, existen conexiones laterales de inhibición (peso negativo) implícitas, a pesar de no estar conectadas, cada una de estas neuronas va a tener cierta influencia sobre sus vecinas. El valor que se asigne a los pesos de las conexiones feedforward entre las capas de entrada y salida (w_{ij}) durante el proceso de aprendizaje de la red va a depender precisamente de esta interacción lateral.



Estructura de una red de Kohonen

La influencia que cada neurona ejerce sobre las demás es función de la distancia entre ellas, siendo muy pequeñas cuando están muy alejadas. Es frecuente que dicha influencia tenga la forma de un sombrero mejicano. Se han descubierto conexiones de este tipo entre las neuronas del sistema nervioso central de los animales. [Redes Competitivas, 2000]



2.3. C4.5

El C4.5 forma parte de la familia de los TDIDT (Top Down Induction Trees), junto con antecesor el ID3. Pertenece a los métodos inductivos del Aprendizaje Automático que aprenden a partir de ejemplos preclasificados. Se utilizan en Minería de datos para modelar las clasificaciones en los datos mediante árboles de decisión. Tanto el ID3 como el C4.5 fueron propuestos por Quinlan, el primero en la década de los ochenta y el segundo en 1993. El C4.5 es una extensión del ID3, que sólo trabaja con valores discretos en los atributos. El C4.5, en cambio, permite trabajar con valores continuos, separando los posibles resultados en dos ramas: una para aquellos A_i<=N y otra para A_i>N. Se genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente, aplicando la estrategia de profundidad-primero *(depth-first)*. 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. 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.

Estos algoritmos han tenido gran impacto en la Minería de Datos. Forman parte del grupo de sistemas de aprendizaje supervisado. Han tenido muy buena performance en aplicaciones de dominio médico, artificiales y el análisis de juegos de ajedrez. Posee un nivel alto de precisión en la clasificación, pero no hace uso del conocimiento del dominio. [García Martínez et al, 2003]

3- APLICACIÓN

3.1 Backpropagation

Este algoritmo utiliza una función o superficie de error asociada a la red, buscando el estado estable de mínimo error a través del camino descendente de la superficie de error. Por esta razón es que realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

Funcionamiento del algoritmo.

Dada una neurona (U_i) y su salida y_i, el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad U_j (w_{ij}) para un patrón de aprendizaje p es:

$$\Delta w_{ij}(t+1) = \alpha \delta_{pj} y_{pi}$$

El subíndice p se refiere al patrón de aprendizaje concreto y α es la constante de o tasa de aprendizaje.

La regla delta generalizada difiere con la regla delta en el valor concreto de δ_{pj}. En las redes multinivel con capas ocultas no se conoce la salida deseada de estas capas para poder calcular los pesos en función del error cometido. Sin embargo, inicialmente podemos conocer la salida deseada de las neuronas de salida. Para la unidad U_j de salida, definimos:

$$\delta_{pj} = (d_{pj} - y_{pj}) \square f'\left(net_j\right)$$

donde d_{pj} es la salida deseada para la neurona j y el patrón p y net_j es la entrada neta de la neurona j. Si U_j no es de salida, se tiene:

$$\delta_{pj} = \left(\sum_k \delta_{pk} w_{kj}\right) \square f'\left(net_j\right)$$

donde el rango de k cubre todas las neuronas las que está conectada la salida de U_j. El error que se produce en una neurona oculta es la suma de todos los errores cometidos por las neuronas a las que está conectada su salida, multiplicados por el peso de la conexión correspondiente.

Adición de un momento en la regla delta generalizada.

Se utiliza una amplitud de paso dada por la tasa de aprendizaje α. A mayor tasa, mayor la modificación de los pesos en cada iteración, más rápido será el aprendizaje dando lugar a oscilaciones. Para filtrarlas un término (momento) β,

$$w_{ji}(t+1) = w_{ji}(t) + \alpha \delta_{pj} y_{pi} + \beta(w_{ij}(t) - w_{ji}(t-1)) =$$
$$\Delta w_{ji}(t+1) = \alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t)$$

donde β es una constante que determina el efecto en t+1 del cambio de los pesos ene el instante t.

Esto permite la convergencia de la red en menor número de iteraciones, ya que si en t el incremento de un peso era positivo y en t+1 también, entonces el descenso por la superficie de error en t+1 es mayor. Sin embargo, si en t el incremento era positivo y en t+1 es negativo,

el paso que se da en t+1 es más pequeño, lo que es adecuado ya que significa que se ha pasado por un mínimo y que los pesos deben ser menores para poder alcanzarlo.

Estructura y aprendizaje de la red backpropagation.

Presenta una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa neuronas ocultas internas. Cada neurona (menos en la capa de entrada) recibe entrada de todas las neuronas de la capa previa y genera salida hacia todas las neuronas de la capa siguiente (salvo las de salida). No hay conexiones hacia atrás feedback ni laterales o autor recurrentes.

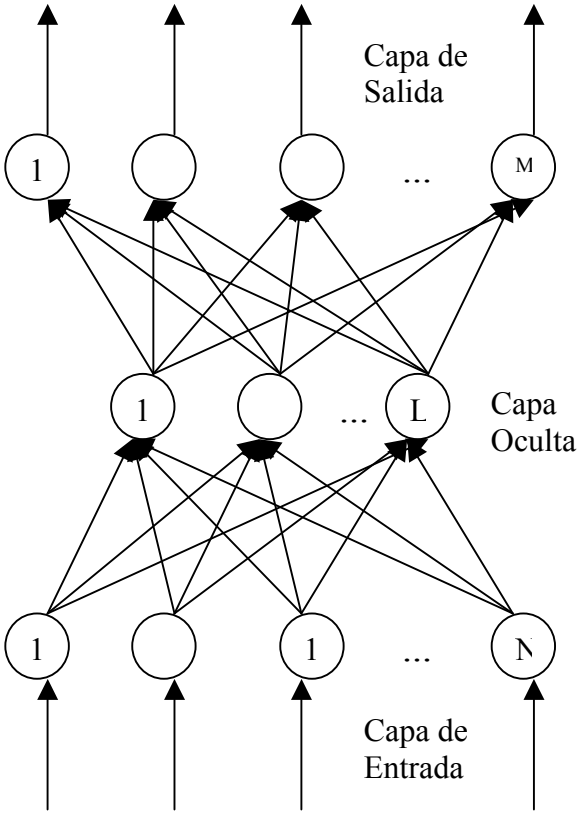


Figura 2.10. Arquitectura de una red Backpropagation

Pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento:

Paso1: Se inicializan los pesos de la red con valores pequeños y aleatorios.

Paso 2: Se presenta un patrón de entrada: Xp=xp1,xp2,...,xpn y se especifica la salida deseada: d1,d2,...,dM (si se utiliza como clasificador, todas las salidas deseadas serán 0, salvo una, que la de la clase a la que pertenece el patrón de entrada).

Paso 3: Se calcula la salida actual de la red, para ello se presentan las entrada y se van calculando la salida que presenta cada capa hasta llegar a la capa de salida, y1,y2,...,yM. Luego

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.
- Para una neurona j oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ij}^{h} x_{pi} + \Theta_{j}^{h}$$

donde el índice h se refiere a magnitudes de la capa oculta (hidden), el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_{j}^{h}(net_{pj}^{h})$$

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (capa o: output)

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{jk}^{o} y_{pj} + \Theta_{k}^{o}$$

$$y_{pk} = f_{k}^{o}(net_{pk}^{o})$$

Paso 4: Calcula los términos de error para todas las neuronas.
Si la neurona k es una neurona de la capa de salida, el valor de delta es:

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) f_{k}^{o'}(net_{pk}^{o})$$

La función f debe ser derivable, para ello existen dos funciones que pueden servir: la función lineal (fk(netjk)=netjk) y la función sigmoidal:

$$f_{k}(net_{jk}) = \frac{1}{1 + e^{-net_{jk}}}$$

La elección de la función de salida depende de la forma de representar los datos: si se desean salidas binarias, se emplea la función sigmoidal, en otro caso es tan aplicable una como la otra.

Para la función lineal tenemos $f_{k}^{o'} = 1$, mientras que la derivada de una función sigmoidal es:

$$f_{k}^{o'} = f_{k}^{o}(1 - f_{k}^{o}) = y_{pk}(1 - y_{pk})$$

por lo que el término de error para las neuronas de salida queda:

$$\delta_{pk}^{o} = d_{pk} - y_{pk}$$

para la salida lineal, y

$$\delta_{pk}^{o} = (d_{pk} - y_{pk}) y_{pk}(1 - y_{pk})$$

para la salida sigmoidal.
Si la neurona j no es de salida se tiene:

$$\delta_{pj}^{h} = f_{j}^{h'}(netJ_{pj}^{h}) \sum_{k} \delta_{pk}^{o} w_{jk}^{o}$$

Donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. Para la función sigmoidal:

$$\delta_{pj}^{h} = x_{pi}(1 - x_{pi})\sum_{k}\delta_{pk}^{o} w_{jk}^{o}$$

Donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de ésta, multiplicados por el peso de la conexión.

Paso 5: Actualización de los pesos.

Se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la forma siguiente:

Para la capa de salida:

$$w_{jk}^{o}(t+1) = w_{jk}^{o}(t) + \Delta w_{jk}^{o}(t+1);$$
$$\Delta w_{jk}^{o}(t+1) = \alpha\delta_{pk}^{o} y_{pj}$$

y para los pesos de las neuronas de la capa oculta:

$$w_{ij}^{h}(t+1) = w_{ij}^{h}(t) + \Delta w_{ij}^{h}(t+1);$$
$$\Delta w_{ij}^{h}(t+1) = \alpha\delta_{pj}^{h} x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor

$$\beta(w_{jk}^{o}(t) - w_{jk}^{o}(t-1))$$

en el caso de la neurona de salida, y

$$\beta(w_{ij}^{h}(t) - w_{ij}^{h}(t-1))$$

en el caso de una neurona oculta.

Paso 6: El proceso se repite hasta que el término de error

$$E_{p} = \frac{1}{2}\sum_{k=1}^{M}\delta_{pk}^{2}$$

resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

[García Martínez et al, 2003]

3.2 SOM

Funcionamiento de una red de Kohonen.

Cuando se presenta a la entrada una información cada una de las M neuronas de la capa de salida la recibe a través de las conexiones feedforward con pesos w_{ij}.

$$E_K = \left(e_1^{(k)} \quad e_2^{(k)} \quad \ldots \quad e_N^{(k)} \right)$$

También estas neuronas reciben las entradas producto de las interacciones laterales con el resto de las neuronas de salida y cuya influencia dependerá de las distancia a la que se encuentren. Así, la salida generada por una neurona de salida j ante una vector de entrada E_K será:

$$s_j(t+1) = f\left(\sum_{i=1}^{N} w_{ij} e_i^{(k)} + \sum_{p=1}^{M} Int_{pj} s_p(t) \right)$$

Int_{pj} es una función del tipo sombrero mejicano que representa la influencia lateral de la neurona p sobre la neurona j. La función de activación de las neuronas de salida (f) será del tipo continuo, lineal o sigmoidal, ya que esta red trabaja con valores reales.

SOM es una red de tipo competitivo, ya que al presentarse una entrada E_k, la red evoluciona hasta alcanzar un estado estable en el que solo hay una neurona activada, la ganadora. La formulación matemática del funcionamiento de esta red puede simplificarse así:

$$s_j = \begin{cases} 1 & MIN \left\| E_K - W_J \right\| = MIN \left(\sqrt{\sum_{i=1}^{N} \left(e_i^{(k)} - w_{ij} \right)^2} \right) \\ 0 & resto \end{cases}$$

donde $\left\| E_K - W_J \right\|$ es una medida de la diferencia entre el vector de entrada y el vector de pesos de las conexiones que llegan a la neurona j desde la entrada. Es en estos pesos donde se registran los datos almacenados por la red durante el aprendizaje. Durante el funcionamiento, lo que se pretende es encontrar el dato aprendido más parecido al de entrada para averiguar qué neurona se activará y en que zona del espacio bidimensional de salida se encuentra.

Esta red realiza una tarea de clasificación ya que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información; ante otra entrada parecida se activa la misma neurona o una cercana a la anterior, garantizando que las neuronas topológicamente cercanas sean sensibles a entradas físicamente similares. Por esto, la red es muy útil para establecer relaciones antes desconocidas entre conjuntos de datos.

Aprendizaje

El aprendizaje es de tipo OFF LINE, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. En la primera se fijan los pesos de las conexiones feedforward entra las capas de entrada y salida. Emplea un aprendizaje no supervisado de tipo competitivo. Las neuronas de la capa de salida compiten por activarse y sólo una de ellas permanece activa ante una entrada determinada. Los pesos de las conexiones se ajustan en función de la neurona que haya resultado vencedora.

En el entrenamiento, se presenta a la red un conjunto de informaciones de entrada para que ésta establezca las diferentes clases que servirán durante la fase de funcionamiento para realizar la clasificación de los nuevos datos que se presenten. Los valores finales de los pesos de las conexiones feedforward que llegan a cada neurona de salida se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente. Si existiesen más vectores de entrenamiento que neuronas de salida, más de un vector deberá asociarse a la misma clase. En tal caso, los pesos se obtienen como un promedio de dichos patrones.

Durante el entrenamiento habrá que ingresar varias veces todo el juego de entrenamiento para refinar el mapa topológico de salida consiguiendo que la red pueda realizar una clasificación más selectiva.

El algoritmo de aprendizaje es el siguiente [Hiler José R., Martínez Víctor J., 1994]:

1) En primer lugar, se inicializan los pesos (w_{ij}) con valores aleatorios pequeños y se fija la zona inicial de vecindad entre las neuronas de salida.
2) A continuación se presenta a la red una información de entrada (la que debe aprender) en forma de vector $E_K = \left(e_1^{(k)} \quad \ldots \quad e_N^{(k)}\right)$, cuyas componentes $e_i^{(k)}$ serán valores continuos.
3) Se determina la neurona vencedora a la salida. Esta será aquella j cuyo vector de pesos W_j sea el más parecido a la información de entrada E_k. Para ello se calculan las distancias entre ambos vectores, una para cada neurona de salida. Suele utilizarse la distancia euclídea o bien la siguiente expresión similar pero sin la raíz:

$$d_j = \sum_{i=1}^{N} \left(e_i^{(k)} - w_{ij}\right)^2 \qquad 1 \le j \le M$$

donde: $e_i^{(k)}$: Componente i-ésimo del vector k-ésimo de entrada.

w_{ij}: Peso de la conexión entra las neuronas i (de entrada) y j (de salida).

4) Una vez localizada la neurona vencedora j*, se actualizan los pesos de las conexiones feedforward que llegan a dicha neurona y a sus vecinas. Con esto se consigue asociar la información de entrada con cierta zona de la capa de salida.

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\left[e_i^{(k)} - w_{ij*}(t)\right] \qquad j \in Zona_{j*}(t)$$

$Zona_{j*}(t)$ es la zona de vecindad de la neurona vencedora j+. El tamaño de esta zona se puede reducir en cada iteración del entrenamiento aunque en la práctica es habitual mantener esa zona fija.

El término $\alpha(t)$ es el coeficiente de aprendizaje y toma valores entre 0 y 1. Este parámetro decrece con cada iteración. De esta forma, cuando se ha presentado todo el juego de datos un gran número de veces, α tiende a cero y las variaciones de pesos son insignificantes.

$\alpha(t)$ suele tener alguna de las siguientes expresiones: $\alpha(t) = \frac{1}{t}$ $\qquad \alpha(t) = \alpha_1\left(1 - \frac{t}{\alpha_2}\right)$

5) El proceso se repite un mínimo de 500 veces ($t \ge 500$).

[García Martínez et al, 2003]

3.3 C.4.5

Forma parte de la familia de los *Top Down Induction Trees* (TDIDT), pertenecientes a los métodos inductivos del Aprendizaje Automático que aprenden a partir de ejemplos preclasificados. En Minería de Datos, se utiliza para modelar las clasificaciones en los datos mediante árboles de decisión.

Construcción de los árboles de decisión

Los árboles TDIDT, (ID3 y C4.5), se construyen con el método de Hunt. Se parte de un conjunto T de datos de entrenamiento. Dadas las clases $\{C_1, C_2, \ldots, C_k\}$, existen tres posibilidades:

1. T contiene uno o más casos, todos pertenecientes a un única clase C_j: El árbol de decisión para T es una hoja identificando la clase C_j .

2. T no contiene ningún caso: El árbol de decisión es una hoja, pero la clase asociada debe ser determinada por información que no pertenece a T. Por ejemplo, una hoja puede escogerse de acuerdo a conocimientos de base del dominio, como ser la clase mayoritaria.

3. T contiene casos pertenecientes a varias clases: Se refina T en subconjuntos de casos que tiendan hacia una colección de casos de una única clase. Se elige una prueba basada en un único atributo, que tiene uno o más resultados, mutuamente excluyentes $\{O_1, O_2, \ldots, O_n\}$. T se particiona en los subconjuntos $T_1, T_2, \ldots, T_n$ donde T_i contiene todos los casos de T que tienen el resultado O_i para la prueba elegida. El árbol de decisión para T consiste en un nodo de decisión identificando la prueba, con una rama para cada resultado posible. El mecanismo de construcción del árbol se aplica recursivamente a cada subconjunto de datos de entrenamientos, para que la i-ésima rama lleve al árbol de decisión construido por el subconjunto T_i de datos de entrenamiento.

A continuación se presenta el algoritmo del método ID3 (antecesor del C4.5) para la construcción de árboles de decisión en función de un conjunto de datos previamente clasificados.

Función ID3: (R: conjunto de atributos no clasificadores, C: atributo clasificador, S: conjunto de entrenamiento) devuelve un árbol de decisión;

Comienzo

Si S está vacío,
devolver un único nodo con Valor Falla;
Si todos los registros de S tienen el mismo valor para el atributo clasificador,
Devolver un único nodo con dicho valor;
Si R está vacío, entonces
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, entonces
D ← atributo con mayor Ganancia (D, S) entre los atributos de R;
Sean {dj| j=1,2, .., m} los valores del atributo D;
Sean {Sj| j=1,2, .., m} los subconjuntos de S correspondientes a los valores de dj respectivamente;
Devolver un árbol con la raíz nombrada como D y con los arcos nombrados d1, d2, .., dm que van respectivamente a los árboles
ID3(R-{D}, C, S1), ID3(R-{D}, C, S2), .., ID3(R-{D}, C, Sm);
Fin

Cálculo de la Ganancia de Información

Cuando los casos en un conjunto *T* contiene ejemplos pertenecientes a distintas clases, se realiza una prueba sobre los distintos atributos y se realiza una partición según el "mejor" atributo. Para encontrar el "mejor" atributo, se utiliza la teoría de la información, que sostiene que la información se maximiza cuando la entropía se minimiza. La entropía determina la azarosidad o desestructuración de un conjunto.

Supongamos que tenemos ejemplos positivos y negativos. En este contexto la entropía de un subconjunto S_i, $H(S_i)$, puede calcularse como:

$$H(S_i) = -p_i^+ \log p_i^+ - p_i^- \log p_i^-$$

Donde p_i^+ es la probabilidad de que un ejemplo tomado al azar de S_i sea positivo. Esta probabilidad puede calcularse como

$$p_i^+ = \frac{n_i^+}{n_i^+ + n_i^-} \qquad \textbf{(2.1)}$$

Si el atributo *at* divide el conjunto *S* en los subconjuntos S_i, $i = 1,2, \ldots\ldots, n$, entonces, la entropía total del sistema de subconjuntos será:

$$H(S,at) = \sum_{i=1}^{n} P(S_i) \cdot H(S_i) \qquad \textbf{(2.2)}$$

Donde $H(S_i)$ es la entropía del subconjunto S_i y $P(S_i)$ es la probabilidad de que un ejemplo pertenezca a S_i. Puede calcularse, utilizando los tamaños relativos de los subconjuntos, como:

$$P(S_i) = \frac{|S_i|}{|S|} \qquad \textbf{(2.3)}$$

La ganancia en información puede calcularse como la disminución en entropía. Es decir:

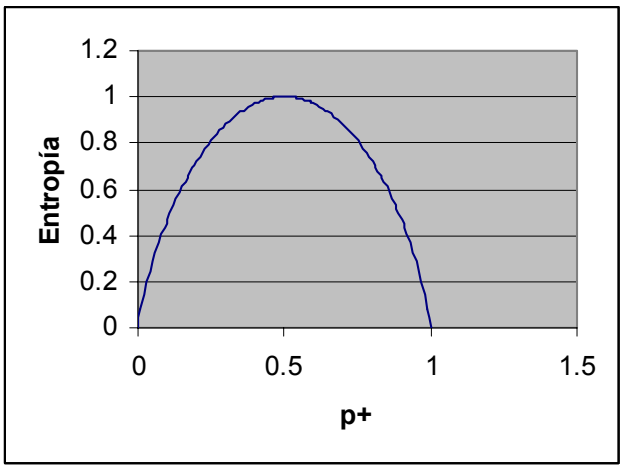
$$I(S,at) = H(S) - H(S,at) \qquad \textbf{(2.4)}$$

Donde $H(S)$ es el valor de la entropía a priori, antes de realizar la subdivisión, y $H(S,at)$ es el valor de la entropía del sistema de subconjuntos generados por la partición según *at*.

El uso de la entropía para evaluar el mejor atributo no es el único método existente o utilizado en *Aprendizaje Automático*. Sin embargo, es el utilizado por Quinlan al desarrollar el ID3 y su sucesor el C4.5.

Entropía

Es la cantidad de información que se espera observar cuando un evento ocurre según una distribución de probabilidades. Mide la incertidumbre dada una distribución de probabilidades. Si tomamos un conjunto con elementos positivos y negativos, la entropía variará entre 0 y 1.



Es 0 si todos los ejemplos pertenecen a la misma clase, y 1 cuando hay igual número de ejemplos positivos y negativos en el conjunto de datos. Cuanto tenemos *c* clases posibles, el valor máximo de la entropía será $log_2 c$.

La probabilidad de que un ejemplo tomado al azar pertenezca a la clase *i* y se calcule en base a la frecuencia de los datos de dicha clase en los datos de entrenamiento, se representa en la siguiente fórmula

$$H(S_i) = \sum_{i=1}^{n} - p_i \log p_i$$

Proporción de ganancia

Favorece a los atributos que tienen muchos valores frente a los que tienen pocos valores. Si se tiene un conjunto de registros con fecha y se particiona según el campo fecha, se obtendrá un árbol perfecto, pero que no servirá para clasificar casos futuros, dato el gran tamaño del mismo. Para resolver esta situación se divide a los datos de entrenamiento en conjuntos pequeños, con lo cual tendrá una alta ganancia de información.

Una alternativa para dividir a los datos es la *ganancia de información*. Esta medida penaliza a los atributos como fecha al incorporar el término de información de la división

$$I\_división(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right)$$

La información de la división no es otra cosa que la entropía del conjunto con respecto al atributo *i*. Se define, entonces, a la proporción de ganancia como:

$$proporción\_de\_ganancia(X) = \frac{I(T,X)}{I\_división(X)}$$

La información de la división penalizará a aquellos atributos con muchos valores uniformemente distribuidos. Si tenemos *n* datos separados perfectamente por un atributo, la

información de la división para ese caso será $\log_2 n$. En cambio, un atributo que divide a los ejemplos en dos mitades, tendrá una información de la división de 1.

Cuando la información de la división es cercana a cero, pueden aplicarse varias heurísticas. Puede utilizarse la ganancia como medida y utilizar la proporción de ganancia sólo para los atributos que estén sobre el promedio.

Datos Numéricos

Cuando los árboles de decisión se generan con atributos discretos, la partición del conjunto según el valor de un atributo es simple. Por ejemplo, agrupamos todos los animales que tengan pico, siendo *tiene_pico* un atributo y sus posibles valores *si* y *no.* Cuando los atributos son continuos, no es tan fácil. Por ejemplo, si queremos partir los días de un mes en función a la cantidad de lluvia caída, es casi imposible que encontremos dos días con exactamente la misma cantidad de precipitaciones caídas. Para ello se aplica *binarización.* Este método consiste en formar dos rangos de valores de acuerdo al valor de un atributo, que pueden tomarse como simbólicos. Por ejemplo, si en un día hubo 100ml de lluvia, pueden crearse los intervalos [0,100) y [100, $+\infty$) y el cálculo de la entropía se realiza como si los dos intervalos fueran los dos valores simbólicos que puede tomar el atributo.

Poda de los árboles generados

Hay varias razones para podar los árboles generados por los métodos de TDIDT, la sobregeneralización, evaluación de atributos poco importantes o significativos; y el gran tamaño del árbol. Ejemplos con ruido, atributos no relevantes, deben podarse ya que sólo agregan niveles en el árbol y no contribuyen a la ganancia de información. Si el árbol es demasiado grande, se dificulta la interpretación, con lo cual hubiera sido lo mismo utilizar un método de caja negra.

Existen dos enfoques para podar árboles: la pre-poda *(preprunning)*, detiene el crecimiento del árbol cuando la ganancia de información producida al dividir un conjunto no supera un umbral determinado y la post-poda *(postprunning)*, se aplica sobre algunas ramas una vez que se ha terminado.

La pre-poda, no pierde tiempo en construir una estructura que luego será simplificada en el árbol final, busca la mejor manera de partir el subconjunto y evaluar la partición desde el punto de vista estadístico mediante la teoría de la ganancia de información, reducción de errores, etc. Si esta evaluación es menor que un límite predeterminado, la división se descarta y el árbol para el subconjunto es simplemente la hoja más apropiada. Tiene la desventaja de que no es fácil detener un particionamiento en el momento adecuado, un límite muy alto puede terminar con la partición antes de que los beneficios de particiones subsiguientes parezcan evidentes, mientras que un límite demasiado bajo resulta en una simplificación demasiado leve.

La post-poda, es utilizada por el ID3 y el C4.5. Una vez construido el árbol se procede a su simplificación según los criterios propios de cada uno de los algoritmos.

El Principio de Longitud de Descripción Mínima

El fin de los sistemas de aprendizaje es aprender una "teoría" (árboles o reglas de decisión, por ejemplo) del dominio de los ejemplos, predictiva en el sentido de que es capaz de predecir la clase de nuevos instancias.

El Principio de Longitud de Descripción Mínima (MDL) sostiene que la mejor teoría es aquella que minimiza el tamaño y la cantidad de información necesaria para especificar las excepciones. El MDL provee una forma de medir la performance de los algoritmos basándose en los datos de entrenamiento únicamente. Supongamos que un sistema de

aprendizaje genera una teoría *T*, basada en un conjunto de entrenamiento *E,* y requiere una cierta cantidad de bits *L[T]* para codificar la teoría. Dada la teoría, el conjunto de entrenamiento puede codificarse en una cantidad *L[E/T]* de bits. *L[E/T]* está dada por la función de ganancia de información sumando todos los miembros del conjunto de entrenamiento. La longitud de descripción total de la teoría es *L[E]+L[E/T]*. El principio MDL recomienda la teoría *T* que minimiza esta suma.

Funciones alternativas

La entropía no es la única alternativa para elegir el "mejor" atributo en la partición de datos al momento de construir un árbol de decisión según el método de divide y reinarás, Existen otras medidas alternativas. Una de ellas es la función de pérdida cuadrática: dada una instancia con *k* clases posibles a la que puede pertenecer, el sistema aprendiz devuelve un vector de probabilidades p_1, p_2, ...,p_k de las clases de la instancia. Es decir, p_i indica la probabilidad que tiene la instancia de pertenecer a la clase *i*. Con lo cual, los elementos del vector suman 1.

El resultado verdadero de la clasificación de la instancia será una de las clases posibles, entonces, si lo expresamos en un vector a_1, a_2, ...,a_k donde a_i=1 si el elemento es de clase *i* y es 0 en caso contrario.

Entonces, utilizamos la siguiente función para evaluar la pérdida de información según cada atributo:

Ejemplo

$$\sum_{j=1}^{k}\left(p_j - a_j\right)^2 = 1 + 2p_i + \sum_{j=1}^{k} {p_j}^2$$

Atributos Desconocidos

El método de Hunt, considera los resultados de todas las pruebas para todos los casos conocidos. Pero cuando los datos están incompletos, podemos tomar dos caminos posibles: descartar una proporción importante de los datos por incompletos y declarar algunos casos como inclasificables, o adaptar los algoritmos para poder trabajar con valores de atributos faltantes. La primera opción es inaceptable. Para la segunda opción, hay tres cuestiones importantes que deben ser tenidas en cuenta:

1. Selección de una prueba en la cual la partición del conjunto de entrenamiento se realiza en base a un criterio heurístico como ser la *ganancia* o la *proporción de ganancia*. Si dos pruebas distintas utilizan atributos con distinta cantidad de valores desconocidos, ¿cómo debe tenerse esto en cuenta al medir su importancia relativa?
2. Una vez que una prueba ha sido seleccionada, los casos de entrenamiento con valores desconocidos para los atributos relevantes no pueden ser asociados con una respuesta particular de la prueba, y, por lo tanto, no pueden asignarse a un subconjunto $\{T_i\}$. ¿Cómo deben tratarse estos casos durante la partición?
3. Cuando el árbol de decisión se utiliza para clasificar un caso nuevo, ¿cómo debe proceder el sistema al encontrarse con un valor de atributo desconocido para el nodo de decisión que está tratando de evaluar?

Varios autores han tratado de resolver estos problemas, generalmente rellenando los valores desconocidos con los valores más frecuentes. En un estudio realizado por Quinlan, se comparan las soluciones más comunes a este problema; y se llega a la conclusión de que existen varios enfoques que son notablemente inferiores, pero no existe ningún enfoque que sea claramente superior.

A continuación se presenta un resumen del estudio.

Estudio sobre datos con atributos desconocidos en la Inducción

Métodos analizados

Todos los enfoques que a continuación se describen fueron implementados como variantes de un programa que construye un árbol de decisión utilizando la *proporción de ganancia*. Los árboles producidos no fueron podados. Varios enfoques para solucionar los tres problemas fueron explorados. Cada uno de ellos tiene una letra identificatoria, tal que una combinación de letras implica una combinación de métodos.

Al evaluar una prueba basada en el atributo *A*

- I - Ignorar los casos del conjunto de entrenamiento con valores desconocidos
- R - Reducir la ganancia de información aparente al testear *A* en la proporción de casos con valores desconocidos para A: si *A* tiene una proporción de valores desconocidos del *x%*, la prueba sobre *A* no dará información *x%* del tiempo.
- S - "Completar" los valores desconocidos de *A* antes de calcular la *ganancia* de *A*, basándose en los valores de otros atributos
- C - Completar los valores de *A* con el valor más frecuente para el atributo antes de calcular la *ganancia.*

Al partir el conjunto de entrenamiento utilizando una prueba sobre el atributo *A* y un caso de entrenamiento tiene un valor desconocido de *A*.

- I - Ignorar el caso
- S - Determinar el valor de *A* utilizando el método de Shapiro y asignarlo al subconjunto correspondiente.
- C - Tratar el caso como si tuviera el valor más común de *A*.
- P - Asignar el caso a uno de los subconjuntos con probabilidad proporcional al número de casos con valores conocidos en cada subconjunto.
- F - Asignar una fracción del caso a cada subconjunto utilizando las proporciones explicadas en el inciso anterior.
- A - Incluir el caso en todos los subconjuntos
- U - Desarrollar una rama separada para los casos de valores desconocidos de *A*.

Al clasificar un caso nuevo con un valor desconocido del atributo *A* que debe ser evaluado.

- U - Si existe una rama especial para los valores desconocidos de *A*, tomarla
- S - Determinar el resultado más probable de *A* y actuar de acuerdo con ello.
- C - Tratar el caso como si fuese el del valor más común de *A*.
- F - Explorar todas las ramas, combinando los resultados para reflejar las probabilidades de los distintos resultados.
- H - Parar en este punto y asignar el caso a la clase más frecuente.

Casos analizados

- *Valores desconocidos al particionar:* los resultados de las pruebas revelan una clara superioridad del RFF (asignar casos fraccionales a los subconjuntos) y una clara desventaja del RIF (ignorar los casos de entrenamiento con valores desconocidos).
- *Valores desconocidos al clasificar:* la estrategia de parar ante los valores desconocidos dio muy malos resultados, mientras que todos las otras estrategias dieron resultados similares
- *Valores desconocidos al seleccionar las pruebas:* ignorar los valores desconocidos dio resultados peores que reducir la ganancia o completar los valores, pero no existió un método claramente superior entre estos dos últimos.

Resultados obtenidos

El estudio se concentró en dominios con altos niveles de valores desconocidos y conjuntos de entrenamiento chicos. Este estudio proporcionó evidencia para las siguientes hipótesis:

- En la evaluación de pruebas, los enfoques que ignoran los casos con valores desconocidos presentan malos resultados cuando esta proporción varía de atributo en atributo.
- Cuando el conjunto de entrenamiento se parte ignorando los casos con valores desconocidos para el atributo probado, se obtienen resultados pobres. El enfoque de dividir los casos entre los subconjuntos resultó muy bueno.

Durante la clasificación, tratar de determinar el resultado más probable de una prueba, funciona bien en algunos dominios, pero muy mal en otros. La combinación de todos los resultados posibles es más resistente, dando una mayor certeza en la clasificación general.

Transformación a Reglas de Decisión

Los árboles de decisión demasiado grandes son difíciles de entender porque cada nodo debe ser interpretado dentro del contexto fijado por las ramas anteriores. Cada prueba tiene sentido si se analiza junto con los resultados de las pruebas previas. Cada prueba tiene un contexto único. Puede ser muy difícil comprender un árbol en el cual el contexto cambia demasiado seguido al recorrerlo. Además, la estructura puede hacer que un concepto en particular quede fragmentado, lo cual hace que el árbol sea aún más difícil de entender. Existen dos maneras de solucionar estos problemas: definir nuevos atributos que estén relacionados con las tareas o cambiar de método de representación, por ejemplo, a reglas de decisión.

En cualquier árbol de decisión, las condiciones que deben satisfacerse cuando un caso se clasifica por una hoja pueden encontrarse analizando los resultados de las pruebas en el camino recorrido desde la raíz. Es más, si el camino fuese transformado directamente en una regla de producción, dicha regla podría ser expresada como una conjunción de todas las condiciones que deben ser satisfechas para llegar a la hoja. Consecuentemente, todos los antecedentes de las reglas generadas de esta manera serían mutuamente excluyentes y exhaustivos. Al hablar de reglas de decisión o de producción nos referimos a una estructura de la forma:

Si $atributo_1=valor_X$ **y** $atributo_2=valor_Y$ .... **y** $atributo_n=valor_Z$ **Entonces** $clase_K$

Resolución de un ejemplo utilizando el ID3

Se presentarán un árbol y un conjunto de reglas de decisión obtenidos utilizando ID3. Supongamos que queremos analizar cuáles días son convenientes para jugar al tenis basándonos en la humedad, el viento y el estado del tiempo. Los datos se presentan en la siguiente tabla:

Estado	Humedad	Viento	JuegoTenis
Soleado	Alta	Leve	No
Soleado	Alta	Fuerte	No
Nublado	Alta	Leve	Si
Lluvia	Alta	Leve	Si
Lluvia	Normal	Leve	Si
Lluvia	Normal	Fuerte	No
Nublado	Normal	Fuerte	Si
Soleado	Alta	Leve	No
Soleado	Normal	Leve	Si
Lluvia	Normal	Leve	Si
Soleado	Normal	Fuerte	Si
Nublado	Alta	Fuerte	Si
Nublado	Normal	Leve	Si
Lluvia	Alta	Fuerte	Si

Los árboles y las reglas obtenidos utilizando la ganancia y la proporción de ganancia son iguales.

Construcción del árbol de decisión

A partir de todos los datos disponibles, el ID3 analiza todas las divisiones posibles según los distintos atributos y calcula la ganancia y/o la proporción de ganancia. Comecemos analizando el atributo Estado.

El atributo Estado tiene la siguiente distribución de datos:

	Lluvia	Nublado	Soleado
No	1	0	3
Si	4	4	2
Totales	5	4	5

Para calcular la ganancia y, por lo tanto, también la proporción de ganancia, es necesario calcular la entropía del conjunto. Entonces,

$$H(S) = -p^{Si}\log_2 p^{Si} - p^{No}\log_2 p^{No} = -\frac{10}{14}\log_2\frac{10}{14} - \frac{4}{14}\log_2\frac{4}{14} = 0.86312 bits$$

Calculamos ahora la entropía que tendrían los conjuntos resultantes de la división de datos según este atributo.

$$H(S, Estado) = \sum_{i=1}^{2} P(S_i)\cdot H(S_i) = \frac{5}{14}\left(-\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5}\right) + \frac{4}{14}\left(-\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

$$H(S, Estado) = \frac{5}{14}\times 0.7219 + \frac{4}{14}\times 0 + \frac{5}{14}0.97095 = 0.6046 bits$$

Ahora calculamos la ganancia resultante de dividir al subconjunto según el atributo Estado, tendremos:

$$Ganancia(S, Estado) = H(S) - H(S, Estado) = 0.25852 bits$$

Para calcular la proporción de ganancia debemos conocer primero la información de la división que se calcula como:

$$I\_división(S) = -\sum_{i=1}^{n}\frac{|S_i|}{|S|}\times\log_2\left(\frac{|S_i|}{|S|}\right) = -\frac{5}{14}\times\log_2\left(\frac{5}{14}\right) - \frac{4}{14}\times\log_2\left(\frac{4}{14}\right) - \frac{5}{14}\times\log_2\left(\frac{5}{14}\right) = 1.577 bits$$

Finalmente, calculamos la proporción de ganancia.

$$proporción\_de\_ganancia(S) = \frac{Ganancia(S)}{I\_división(S)} = 0.491042 bits$$

De la misma manera en que calculamos la ganancia y la proporción de ganancia para el caso anterior, calculamos para el atributo Humedad los siguientes valores:

- Ganancia=0.0746702 bits
- Proporción de ganancia =0.14934 bits

Para el caso del atributo Viento obtenemos los siguientes valores:

- Ganancia=0.00597769 bits
- Proporción de ganancia =0.0122457 bits

Una vez que hemos calculado las ganancias y proporciones de ganancia para todos los atributos disponibles, debemos elegir el atributo según el cual dividiremos a este conjunto de datos. Recordemos que tanto en el caso de la ganancia como en el de la proporción de ganancia, el mejor atributo para la división es aquel que la maximiza.

En este ejemplo, la división según el atributo Estado es la que mayor ganancia y proporción de ganancia ofrece. Esto significa que el nodo raíz del árbol será un nodo que evalúa el atributo Estado.

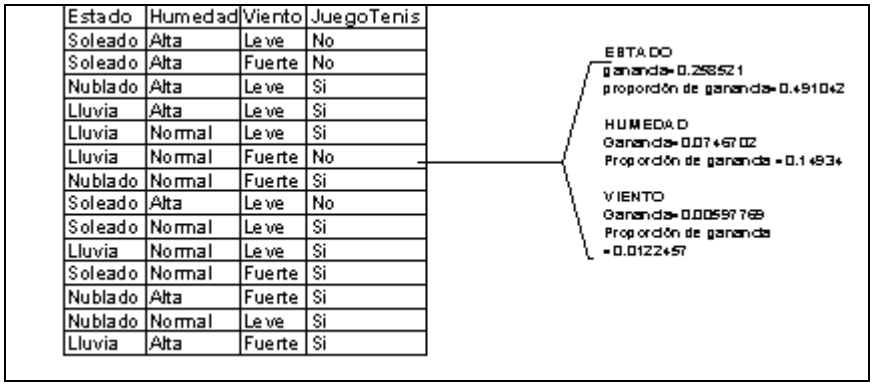


Figura 4.2: Esquema de la construcción de un árbol de decisión utilizando el ID3

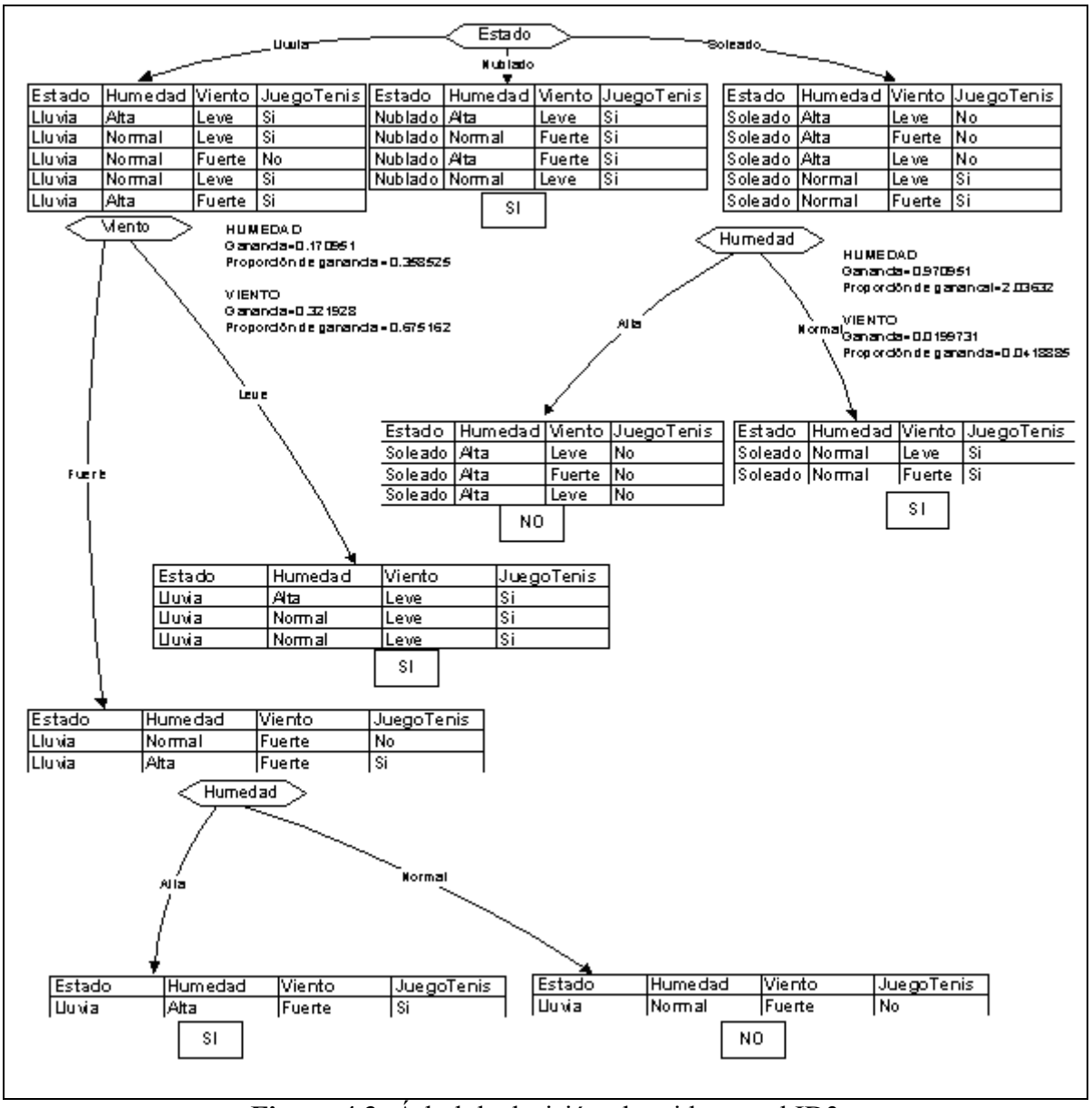


Figura 4.3: Árbol de decisión obtenido con el ID3

Transformación a reglas de decisión

Como se explicó en la sección 4.3 para pasar un árbol de decisión a reglas de decisión, el ID3 lo recorre en preorden y cada vez que llega a una hoja, escribe la regla que tiene como consecuente el valor de la misma, y como antecedente, la conjunción de las pruebas de valor especificados en todos los nodos recorridos desde la raíz para llegar a dicha hoja. Analicemos el pasaje del árbol de la figura 4.3 a reglas de decisión.

El recorrido del árbol comienza por la raíz Estado, continúa por los nodos Viento y Humedad hasta llegar a la hoja "SI". La regla generada para este recorrido será:

Regla 0

SI Estado = Lluvia Y Viento = Fuerte Y Humedad = Alta
ENTONCES JuegoTenis = Si

Si seguimos el recorrido preorden, llegamos a continuación a la hoja "NO", obteniendo en este caso la siguiente regla:

Regla 1

SI Estado = Lluvia Y Viento = Fuerte Y Humedad = Normal
ENTONCES JuegoTenis = No

Recorriendo en este sentido el árbol, el resto de las reglas obtenidas se muestran a continuación.

Regla 2

SI Estado = Lluvia Y Viento = Leve
ENTONCES JuegoTenis = Si

Regla 3

SI Estado = Nublado ENTONCES JuegoTenis = Si

Regla 4

SI Estado = Soleado Y Humedad = Alta
ENTONCES JuegoTenis = No

Regla 5

SI Estado = Soleado Y Humedad = Normal
ENTONCES JuegoTenis = Si

[García Martínez et al, 2003]

4- HERRAMIENTAS PARA LA EXPLOTACIÓN DE DATOS

4.1 Herramienta de aplicación de Backpropagation (NNclass)

[NNclass, 1998]

4.1.1 Funcionamiento de la herramienta

Hoja ReadMe

Presenta una descripción en inglés de las instrucciones para el uso de la herramienta.

Paso 1: Ingresar los datos

(A) Ingresar los datos en la hoja Data, empezando por la celda AC 105
(B) Las observaciones deben ubicarse en filas y las variables en columnas
(C) Elegir sobre cada columna, el tipo apropiado (Omit, Output, Cont, Cat)

- Si no desea que la columna sea sometida al modelo, seleccione = OMIT
- Si desea que la columna sea tratada como categoría, seleccione = CAT
- Si desea que la columna sea tratada como continua, seleccione = CONT
- Si desea que la columna represente la salida, seleccione = OUTPUT

Para clasificar problemas, se requiere una variable de salida. La aplicación, tratará automáticamente a esta columna, como una variable de salida (OUTPUT).
Se puede ingresar como máximo 50 variables de entrada (INPUT) de las cuales 40 pueden ser variables de tipo categoría (CAT). Debe asegurarse que la cantidad de columnas de entrada y las de tipo categoría coincidan con los datos registrados en la hoja UserInput
(D) La hoja de datos no puede contener celdas con datos en blanco
(E) Entradas tipo Continuas:

- Todo dato no numérico ingresado en este tipo no serán considerados como Datos Perdidos
- La aplicación los reemplazará por algún valor de su misma columna

(F) Entrada tipo Categoría

- Toda celda vacía o conteniendo Error de Excel, será considerado como la Dato Perdido
- La aplicación reemplazará a este valor, por el dato de mayor frecuencia dentro de su categoría
- Los nombres de las categorías no detectarán diferencias entre nombre como: GOOD, Good, GoOd, good. Todos serán considerados como misma categoría
- Puede haber como mínimo 2 observaciones en cada categoría. Si hay una sola, se la debe eliminar o renombrarla y ubicarla en otra categoría

Paso 2: Completar datos en la planilla Input

(A) Completar los datos
(B) Considerar que los valores ingresados están dentro de los rangos que impone la aplicación
(C) Presionar en el botón “Build Model” para ejecutar la aplicación

Paso 3: Resultados obtenidos

(A) Un conjunto de Red Neuronal es básicamente un conjunto de pesos distribuidos entre las capas de la red. Al finalizar la aplicación del modelo se obtendrá el conjunto de pesos finales que serán guardados en la hoja "Calc".
(B) La hoja "Output" de este archivo, mostrará los valores de MSE y ARE del entrenamiento y la validación, mientras el entrenamiento está en progreso de ejecución. Los gráficos del entrenamiento y la validación se muestran en la hoja "Output". Estos pueden ser modificados o borrados.
(C) Si en la hoja "UserInput" se aceptó guardar los resultados en otra hoja, la aplicación creará una nueva hoja conteniendo las entradas del modelo, los datos ingresados, y el modelo ajustado. Este archivo podrá ser utilizado para realizar clasificaciones con cualquier dato que se ingrese.

Paso 4: Cuadro de perfiles

Una vez que está construido el modelo se pueden estudiar los perfiles en la hoja "Profiles". Si la variable de clase tenía k categorías, habrán k curvas por cada perfil. Por cada nueva observación, el modelo estará en condiciones de predecir k puntos por cada k categorías de la variable de clase. El modelo podrá predecir la categoría de clase como la de mayor puntaje. Cada uno de estos puntajes son función de su pronóstico. Si se tienen p predicciones, entonces el puntaje para cada categoría es la superficie dimensional de p. Si p es 2 o menor a 2, no es posible mostrar la superficie gráficamente. El cuadro de perfiles es la mejor forma de visualizar la superficie ajustada. Si se va modificando un solo pronóstico entre dos valores y se van dejando todos los demás valores fijos con algún valor preestablecido, obtendremos el cuadro de perfiles, que es realmente una sección representativa de una sola dimensión de la más alta superficie. En la hoja "Profiles" se puede especificar cual pronóstico modificar y que valores de los demás predictores quedarán fijos. Presionando sobre el botón "Create Profile" se podrá generar el perfil. Si el predictor elegido es de tipo "Categorical", entonces la demás información (puntos a generar, valores de inicio y final), serán ignorados y el gráfico mostrará los puntajes de cada categoría del predictor que fuero elegidos para variar.

El cuadro de perfiles permite estudiar las siguientes cuestiones:

1. Dado un predictor X, qué rango de valores estará asociados a la clase 1, clase 2, clase *k*, etc. (Por ejemplo, se puede encontrar que el puntaje para la clase 1 es mayor cuando X es pequeño; cuando X está en su rango medio el puntaje de la clase 2 es el más alto, y para muchos valores altos de X el puntaje de la clase 3 es el más alto, etc.
2. El gráfico de perfiles permite además estudiar la interacción entre predictores. Supongamos que estamos estudiando los perfiles de la clase 1 y dos predictores son X y Z. Supongamos que estamos observando es perfil dejando a Z con su valor fijo en 1 y variando a X entre -10 y 10. Luego dejamos fijo a Z en 2 en lugar de 1 y variamos a X entre -10 y 10. Si la forma de los perfiles es totalmente diferente entonces quiere decir que los predictores X y Z interaccionan. Es decir, que el efecto de X sobre la variable de clase no es la misma para todos los valores de Z. Para estudiar los efectos de X, importa como Z esté configurado.

Paso 5: Pendiente del Gráfico

Una vez que el modelo está construido, se puede estudiar la pendiente del gráfico para una clase de categoría en particular, en la hoja "LiftChart". Este muestra como la pendiente del gráfico está construida. Suponga que estamos observando la pendiente del gráfico de la categoría k . Observamos todos los puntajes (generados por el modelo) para la categoría k . Ordenamos los datos en forma descendente según estos puntajes. Si el modelo es bueno, podemos esperar que las observaciones que recibieron los puntajes más altos, pueden en realidad pertenecer a la categoría k . Por lo que alrededor de las pocas observaciones más altas, el porcentaje de la categoría actual igual a k , será el más alto. Para N más altas observaciones, se computa el % con al categoría k y el % con la categoría *no k.* Se desarrolla el gráfico con estos dos valores, X = *% not-k* versus Y= *% k* para generar la pendiente de la curva.

La pendiente del gráfico aporta la siguiente información:

1. Qué es el poder discriminatorio del modelo. Si al menos queremos capturar, por ejemplo el 90% de las observaciones de la categoría k exactamente usando el modelo, entonces que porcentaje de no-k observaciones el modelo captura por error. La pendiente de la curva siempre comienza en 0%, 0% y termina en 100%, 100%. La diagonal que nace en 0%,0% y termina en 100%,100% debe estar siempre dibujada como referencia. Cuando mejor es el modelo, más alto resultará la pendiente de la curva sobre la línea diagonal. El mejor escenario posible se da cuanto el 100 % pertenece a la categoría k sin captura un *no-k* . En este caso la curva comienza en 0%, 0% crece verticalmente hacia 0%, 100% y luego se dirige horizontalmente hacia 100%, 100%.

Otros ítems a considerar...

Pesos iniciales:

Para entrenar el modelo necesitamos comenzar con un conjunto de valores iniciales para los pesos de la red. Por defecto los pesos iniciales deben ser valores aleatorios entre $-w$ y w, donde w es un número entre 0 y 1, que debe especificarse en la hoja UserInput.

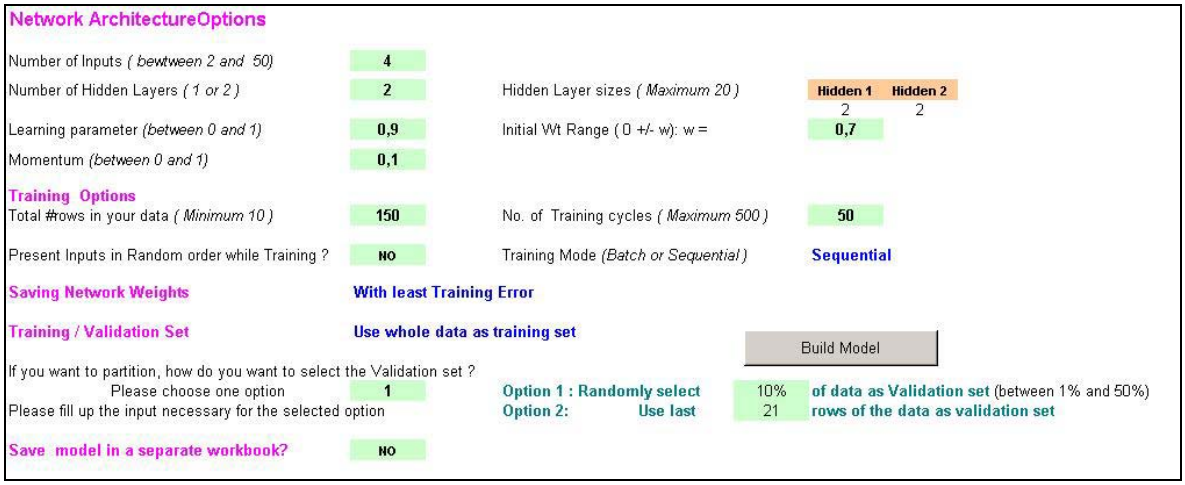
(A) Una vez construido el modelo, los pesos finales son guardados en la hora "Calc". La próxima vez que se quiera entrenar el modelo con la misma arquitectura y los mismos datos, la aplicación preguntará si se quiere comenzar con los pesos guardados en la hoja "Calc". Si se responde que Si, entonces estos pesos será utilizados; pero si se responde No se reinicializarán con valores aleatorios.

(B) En lugar de empezar con pesos aleatorios, se puede empezar con los propios pesos elegidos. Especificar la elección de especificar los pesos, es una tarea no trivial para esta aplicación. Aquí se explica como hacer.

- Especificar las entradas en la hoja "Input" y especificar el número de ciclos de entrenamiento como 0.
- Esto configurará la hoja "Calc" para que no realice el entrenamiento.
- En la hoja "Calc" se debe escribir la elección de pesos en el lugar apropiado de la matriz de pesos.

Se debe volver a la hoja "UserInput" y especificar el número de ciclos deseado y presionar el botón "Build Model". Cuando la aplicación pregunte si se quiere usar los pesos guardados, se debe responder Si. Ahora la red se entrenará con los pesos especificados por el usuario.

Hoja UserInput



En esta hoja se ingresar los parámetros necesarios para configurar el funcionamiento de la aplicación. Estos datos deben ser consistentes con los ingresados en la Hoja "Data". Deben ingresarse:

Opciones de arquitectura de red

- Número de observaciones (entre 2 y 50).
- Número de capas ocultas (entre 1 y 2) – Tamaño de las capas ocultas (Máx. 20)
- Parámetro de aprendizaje (0 ó 1) – Rango inicial de peso Wt (0 +/ -w): w = (valor ingresado)
- Momento (0 ó 1)

Opciones de entrenamiento

- Cantidad total de filas de datos (min. 10) – Número de ciclos de entrenamiento (máx. 50)
- Seleccionar si se presentan entradas en orden aleatorio durante el entrenamiento (Si o No) – Modo de entrenamiento (Serial o Secuencial)

Modo de guardar los pesos de la red Se debe seleccionar entre las siguientes tres opciones:

Al final del último ciclo,
Con el último error validado
Con el último error de entrenamiento

Configuración del entrenamiento y validación. Se debe seleccionar entre las dos alternativas:

Usar una parte de la base de datos
Usar toda la base de datos

Si se desea usar sólo una parte, se debe selecciona el modo de validación. Se puede optar por:

1 (uno) significa que se seleccionará un porcentaje variable de datos a validar (entre 1% y 50%).

2 (dos) significa que se usarán las últimas filas de datos. (Cantidad de filas a ingresar)

Indicar si se quiere guardar los resultados del modelo en una hoja aparte (Si o No)

Hoja Data

Please make sure that there are no more than 50 neurons in Input Layer.
There should be exactly 1 Output variable - application will treat it as Categorical.
There should be no more than 40 Categorical Variables.

Var Type	Omit	Output	Cont	Cont	Cont	Cont	Omit
Var Name	**Species_ID**	**Species_Name**	**Petal_width**	**Petal_length**	**Sepal_width**	**Sepal_length**	**JUNK**
	1	Setosa	2	14	33	50	A
	3	Verginica	24	56	31	67	B
	3	Verginica	23	51	31	69	C
	1	Setosa	2	10	36	46	A
	3	Verginica	20	52	30	65	B
	3	Verginica	19	51	27	58	C
	2	Versicolor	13	45	28	57	A
	2	Versicolor	16	47	33	63	B
	3	Verginica	17	45	25	49	C
	2	Versicolor	14	47	32	70	A
	1	Setosa	2	16	31	48	B
	3	Verginica	19	50	25	63	C
	1	Setosa	1	14	36	49	A
	1	Setosa	2	13	32	44	B
	2	Versicolor	12	40	26	58	C
	3	Verginica	18	49	27	63	A
	2	Versicolor	10	33	23	50	B
	1	Setosa	2	16	38	51	C
	1	Setosa	2	16	30	50	A
	3	Verginica	21	56	28	64	B
	1	Setosa	4	19	38	51	C
	1	Setosa	2	14	30	49	A
	2	Versicolor	10	41	27	58	B
	2	Versicolor	15	45	29	60	C
	1	Setosa	2	14	36	50	A
	3	Verginica	19	51	27	58	B
	1	Setosa	4	15	34	54	C
	3	Verginica	18	55	31	64	A
	2	Versicolor	10	33	24	49	B
	1	Setosa	2	14	42	55	C
	3	Verginica	15	50	22	60	A
	2	Versicolor	14	39	27	52	B

Los datos debe ingresarse a partir de la celda AC105 Los nombres de las variables deben ingresarse en la fila 103. Debe verificarse que la fila 104 esté vacía.

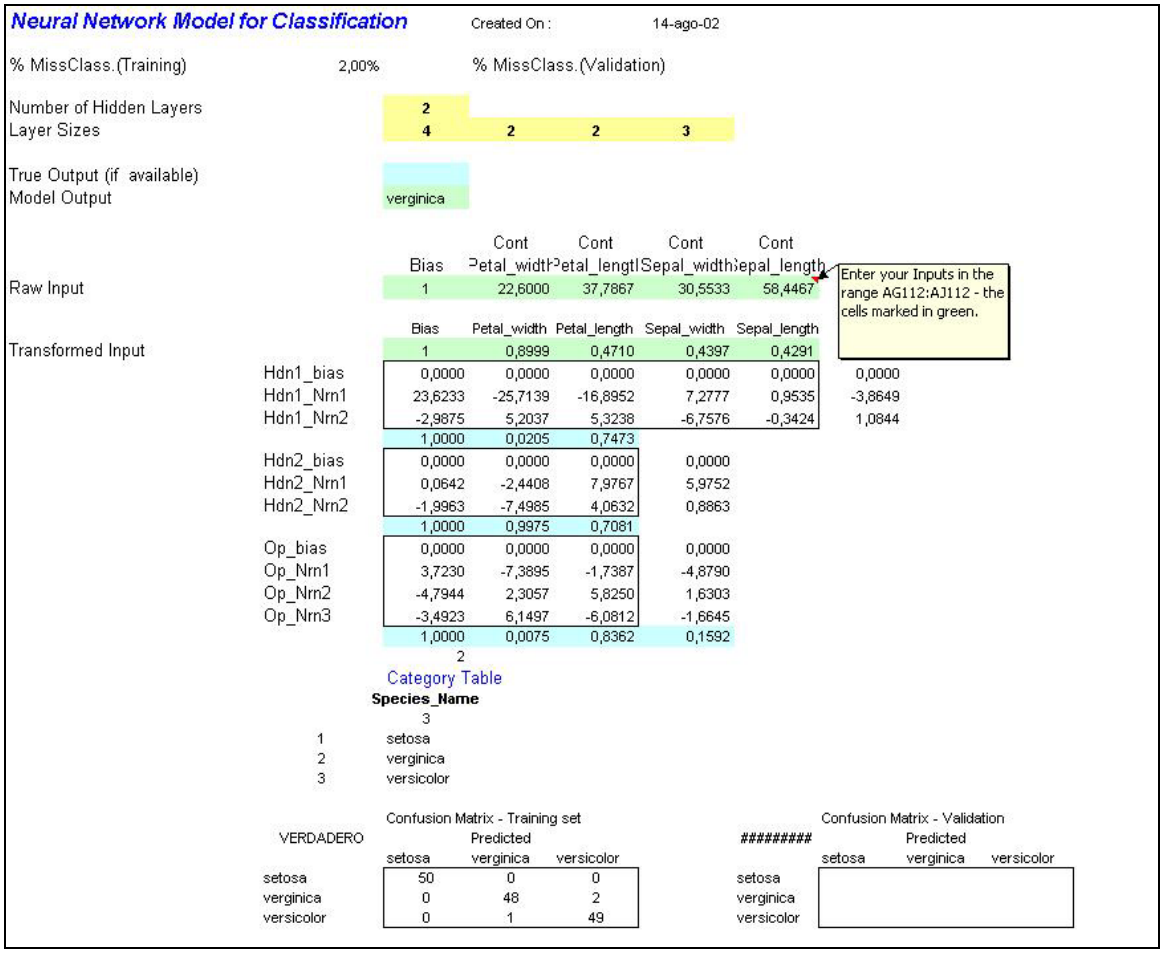
El tipo de variable debe especificarse en la fila 102. CONT (para indicar que la columna es de tipo Continua), CAT (para indicar que una columna es de tipo Categoría), OUTPUT (para dejar la columna fuera de los límites de la aplicación) y OMIT (si no se quiere usar el nombre específico de variable en la fila 103)

Por cada entrada de tipo continua, habrá una neurona en la línea de entrada

Por cada entrada de tipo categórica con k niveles, debe haber k neuronas en la línea de entrada.

No debe haber más de 50 neuronas en la línea de entrada. Debe haber una sola variable de salida, la cual la aplicación la tratará como atributo de tipo categórico. No puede haber más 40 variables de tipo categórico.

Hoja Calc



Neural Network Model for Classification — Created On: 14-ago-02

% MissClass.(Training)	2,00%	% MissClass.(Validation)				
Number of Hidden Layers		2				
Layer Sizes		4	2	2	3	
True Output (if available)						
Model Output		verginica				

	Bias	Cont Petal_width	Cont Petal_length	Cont Sepal_width	Cont Sepal_length	
Raw Input	1	22,6000	37,7867	30,5533	58,4467	Enter your Inputs in the range AG112:AJ112 - the cells marked in green.

	Bias	Petal_width	Petal_length	Sepal_width	Sepal_length	
Transformed Input	1	0,8999	0,4710	0,4397	0,4291	
Hdn1_bias	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Hdn1_Nrn1	23,6233	-25,7139	-16,8952	7,2777	0,9535	-3,8649
Hdn1_Nrn2	-2,9875	5,2037	5,3238	-6,7576	-0,3424	1,0844
	1,0000	0,0205	0,7473			
Hdn2_bias	0,0000	0,0000	0,0000	0,0000		
Hdn2_Nrn1	0,0642	-2,4408	7,9767	5,9752		
Hdn2_Nrn2	-1,9963	-7,4985	4,0632	0,8863		
	1,0000	0,9975	0,7081			
Op_bias	0,0000	0,0000	0,0000	0,0000		
Op_Nrn1	3,7230	-7,3895	-1,7387	-4,8790		
Op_Nrn2	-4,7944	2,3057	5,8250	1,6303		
Op_Nrn3	-3,4923	6,1497	-6,0812	-1,6645		
	1,0000	0,0075	0,8362	0,1592		

2

Category Table

	Species_Name
	3
1	setosa
2	verginica
3	versicolor

Confusion Matrix - Training set

VERDADERO	Predicted setosa	Predicted verginica	Predicted versicolor
setosa	50	0	0
verginica	0	48	2
versicolor	0	1	49

Confusion Matrix - Validation

#########	Predicted setosa	Predicted verginica	Predicted versicolor
setosa			
verginica			
versicolor			

Esta hoja es la que realiza los cálculos del modelo e informa de ellos. Visualiza la siguiente información:

Porcentaje de datos que durante el entrenamiento quedaron fuera de la clasificación y porcentaje de datos que quedaron fuera de la clasificación durante la validación.

Número de capas ocultas seleccionadas previamente

Tamaño de las capas (la de entrada, salida y las ocultas)

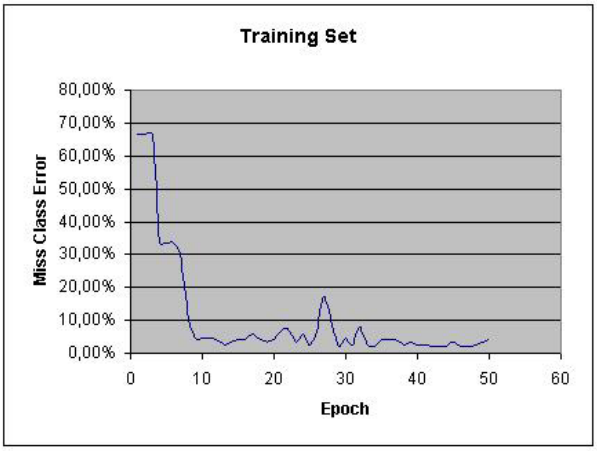
Salidas de tipo verdaderas (en caso de esta disponible)

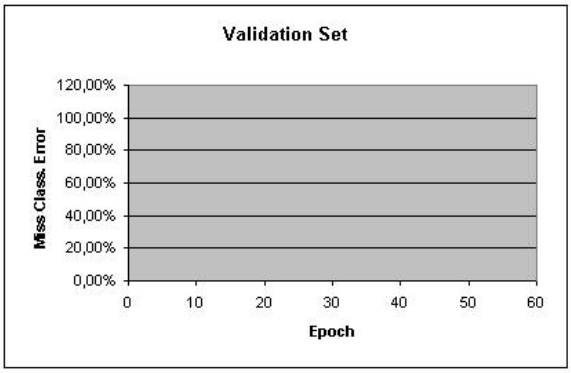
Resultado del modelo. (Este dato se modifica a partir de los cambios en los datos introducidos en las celdas AG112 hasta AJ112)

El resto de los datos dan soporte al modelo para poder dar un resultado.

Hoja Output

Epoch	% Missclassified (Training Set)	% MissClassified (Validation Set)
1	66,67%	
2	66,67%	
3	66,67%	
4	33,33%	
5	33,33%	
6	33,33%	
7	30,00%	
8	10,67%	
9	4,67%	
10	4,67%	
11	4,67%	
12	4,00%	
13	2,67%	
14	3,33%	
15	4,00%	
16	4,00%	
17	6,00%	
18	4,00%	
19	3,33%	
20	4,00%	
21	6,67%	
22	7,33%	
23	3,33%	
24	6,00%	
25	2,67%	
26	6,67%	
27	17,33%	
28	8,67%	
29	2,00%	
30	4,67%	
31	2,67%	
32	8,00%	
33	2,67%	
34	2,00%	
35	4,00%	





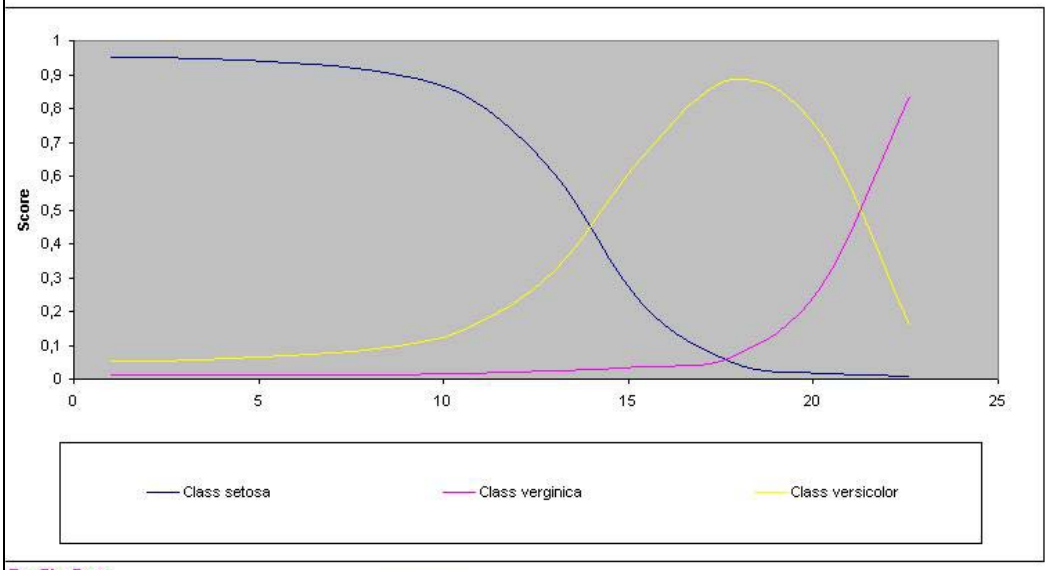
La tabla muestra los porcentajes no incluidos en la clasificación durante el entrenamiento y en la validación, cada una con sus respectivos gráficos. En los resultados obtenidos se observa que a medida que el modelo se entrena, disminuye la cantidad de datos que quedan fuera de la clasificación.

Hoja Profile

Profile plot for the fitted model [Create Profile]

Generate profile for each of the Class categories
Generate 10 data points
by varying **Petal_width** between 1 and 25
keeping the other predictors fixed at the specified values
Notes:
For each of the Class category one profile is created
X axis is the predictor you have chosen to vary
Y axis is the Score - the scaled output of the network for that category
At any value of X - the category having the highest score is the precicted Class category.

Predictor	Petal_width	Petal_length	Sepal_width	Sepal_length
Fixed Value	11,927	37,787	30,553	58,447
Min / Max in Original Data (for user's reference only)				
Min	1,00	10,00	20,00	43,00
Max	25,00	69,00	44,00	79,00

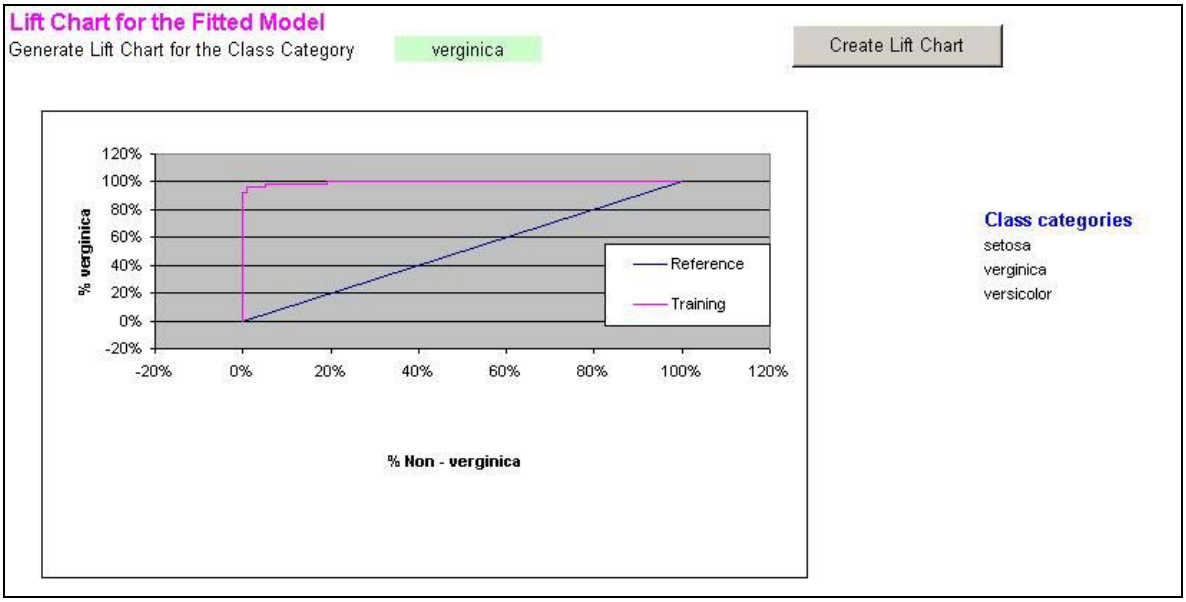


Profile Data
Number of Class Categories 3

Petal_width	Class setosa	Class verginic	Class versicolor
1	0,950806958	0,01039055	0,054206
3,400000095	0,945910009	0,01071688	0,058635
5,800000191	0,935669352	0,01134457	0,06769
8,199999809	0,911019168	0,01264262	0,088612
10,60000038	0,838735225	0,01557428	0,145876
13	0,607454666	0,02258064	0,318859
15,39999962	0,220193285	0,03783236	0,658202
17,79999924	0,049484972	0,06330148	0,885181
20,20000076	0,01703198	0,27544025	0,728232
22,60000038	0,007547487	0,83620432	0,159163

Presionando el botón “Create Profile”, se genera el perfil ajustado del modelo. Para ello se deben ingresar algunos valores, que este proceso tiene en cuenta:

Hoja LifChart



Genera el gráfico para la clase seleccionada.

4.1.2 Bases de Datos de ejemplo

Esta base de datos [Contraceptive Method Choice; 1987] fue realizada por la República de Indonesia en 1987. Los datos fueron obtenidos de mujeres casadas que no estaban embarazadas o lo desconocían, al momento de la entrevista. El objetivo de estudio es poder predecir que tipo de método anticonceptivo eligen las mujeres en función de ciertos datos demográficos y ciertas características socio-económicas.

Una vez entrenada la base de datos lo que se pretende es ingresar valores en cada no de los ítems evaluados, para obtener el método anticonceptivo que usaría dicha mujer.

Descripción de los atributos

- Edad esposa (dato tipo numérico)
- Nivel educativo de esposa (dato tipo categórico) de 1 = bajo, 2, 3, hasta 4= alto
- Nivel educativo de esposo (dato tipo categórico) de 1 = bajo, 2, 3, hasta 4 = alto
- Cantidad de hijos nacidos (dato tipo numérico)
- Religión de esposa (binario) 0 = No-Islámico, 1 = Islámico
- Esposa trabaja? (binario) 0 = Sí ó 1= No
- Ocupación del esposo (dato tipo categórico) 1, 2, 3, 4
- Estándar de vida (dato tipo categórico) 1 = bajo, 2, 3, 4 = alto
- Exposición media (dato tipo binario) 0 = Bueno ó 1 = No Bueno
- Método anticonceptivo (atributo de clase) 1 = (no usa) 2 = métodos de largo plazo, 3 = métodos de corto plazo.

Edad esposa	Nivel educativo de esposa	Nivel educativo de esposo	Cantidad hijos nacidos	Religión de esposa	Esposa trabaja?	Ocupación de esposo	Estándar de vida	Exposición media	Método anticonceptivo
24	2	3	3	1	1	2	3	0	1
45	1	3	10	1	1	3	4	0	1
43	2	3	7	1	1	3	4	0	1
42	3	2	9	1	1	3	3	0	1
36	3	3	8	1	1	3	2	0	1
19	4	4	0	1	1	3	3	0	1
38	2	3	6	1	1	3	2	0	1
21	3	3	1	1	0	3	2	0	1
27	2	3	3	1	1	3	4	0	1
45	1	1	8	1	1	2	2	1	1

4.2 Herramienta de aplicación de SOM (NNclust)

[NNclust, 1998]

4.2.1 Funcionamiento de la herramienta

Hoja ReadMe

Presenta una descripción en inglés de las instrucciones para el uso de la herramienta.

Paso 1: Ingresar los datos

(G) Se deben ingresar los datos en la hoja "Data", a partir de la celda 13
(H) Las observaciones deben ubicarse en filas y las variables en columnas.
(I) Por cada columna se debe elegir el tipo apropiado: ("Use" u "Omit")

a. Si se quiere que el proceso de clasificación excluya a alguna columna se debe selecciona la opción = "OMIT"
b. Si se quiere incluir la columna en la clasificación, se debe elegir = "USE"

Se pueden ingresar un máximo de 50 variables de clasificación. El aplicativo automáticamente tratará a todas las variables como continuas.

Debe asegurarse que el número de variables ingresadas en la hoja "Input" sea la misma cantidad de columnas ingresadas en la hoja "Data" de tipo = "USE".

Debe asegurarse que el número de observaciones ingresados en la hoja "Input" sea igual o menor a las filas ingresadas en la hoja "Data".

(J) No puede haber filas o columnas en blanco.
(K) Todas las variables a usar en la clasificación deben estar en formato numérico. Las que no cumplan con este requisito serán consideradas valores perdidos. La aplicación podría reemplazarlas por algún dato de la misma columna.

Paso 2: Ingreso de parámetros en la hoja "Input"

(A) Nótese que SOM es una parrilla conformada por n-neuronas, organizada en n filas y n columnas. Es necesario especificar el valor de n, n debe se mayor o igual a 2 y menor o igual a 10.
(B) Un ciclo consiste en una presentación de todas las observaciones al mapa. Por tal motivo es necesario especificar la cantidad de ciclos, es decir la cantidad de veces que se presentarán las observaciones al mapa neuronal.
(C) En cada ciclo todas las observaciones serán presentadas. El orden de presentación puede ser al azar o en el orden en el que fueron ingresados los datos en la hoja "Data". Se debe seleccionar si se quiere respetar este orden o dejarlo al azar.
(D) Se debe tener presente que el valor final del parámetro de aprendizaje debe ser menor que el valor inicial y ambos valores deben ser mayor o igual a 0 y menor o igual a 1.
(E) Considerar que el valor final de Sigma es menor que el valor inicial y que ambos valores deben ser mayor o igual al 0% y menor o igual al 100%.
(F) A medida que el entrenamiento de la red progresa, ambos parámetros de aprendizaje y de Sigma decrecen desde el valor inicial hacia el valor final, por lo que se debe seleccionar el rango decreciente, ya sea en forma lineal o exponencial

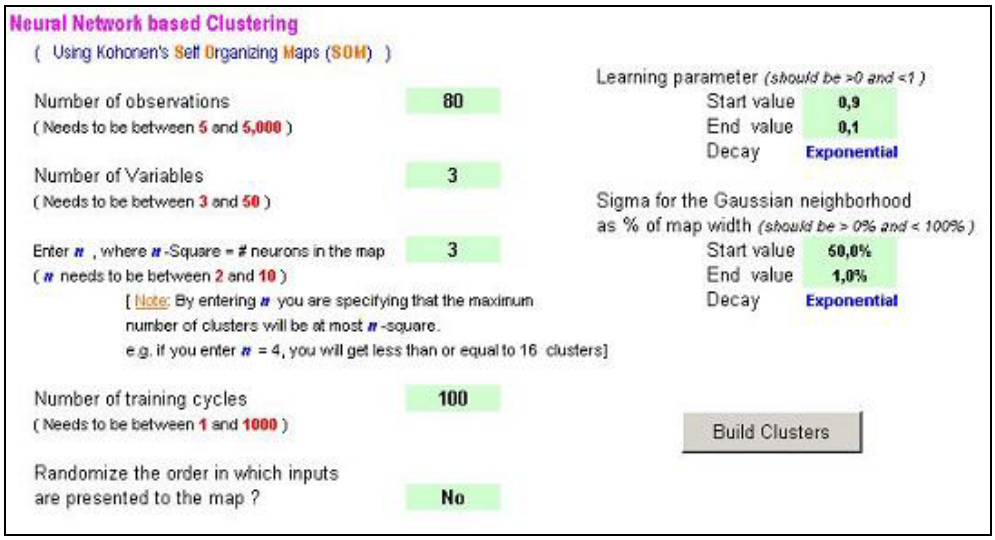
Paso 3: Efecto del botón "Build Clusters"

(A) Mientras el mapa se entrena, los datos de las variables se actualizan de manera que cada valor de las variables se transforman en –1 y 1. Esto es los que se llama normalización de los datos. Este proceso puede ser muy largo, en especial en bases de datos con muchas observaciones y variables.
Si se entrena la red con los mismos datos en dos veces sucesivas, se puede cancelar la normalización en la segunda vuelta. La aplicación preguntará si se quiere cancelar esta normalización o no. Cancelar esta tarea ahorra mucho tiempo. La aplicación siempre se ocupará de chequear el número de filas y columnas en los datos para determinar si ésta ha cambiado desde la última vez que se la ejecutó. No chequea los datos individuales de la hoja "Data". De manera que si se está seguro de que los datos han cambiado desde la última vez que se corrió el algoritmo, se debe nuevamente normalizar los datos.
(B) Si se está entrenando la red con las mismas variables y con las mismas dimensiones de mapa, respecto de la última vez que se corrió, la aplicación preguntará si se quiere comenzar con los pesos obtenidos en el anterior procesamiento. El comenzar con los pesos obtenidos anteriormente aporta incrementos en el aprendizaje. Ésta opción permite resguardar los aprendizajes que se fueron acumulando. Si en cambio los datos se han cambiado, se debe volver a configurar las variables junto con su orden de procesamiento para poder reasignar valores a los pesos.

Paso 4: Resultados de la Clasificación

(A) Los resultados pueden observarse en la hoja "Output". Los datos son de sólo lectura, ya que la planilla está protegida para evitar modificaciones.
(B) La aplicación ofrece la posibilidad de guardar los resultados en una planilla aparte para que el usuario tenga la posibilidad de poder editar sus resultados.
(C) En esta planilla se pueden guardan los datos procesaros por el algoritmo, el cluster asignado a cada observación, y los pesos. Además un gráfico será creado para permitir una comparación visual de los resultados de las variables que atraviesan los diferentes clusters.
(D) En la hoja "Weights", un gráfico dará una representación visual de las observaciones que hay en cada porción del mapa.

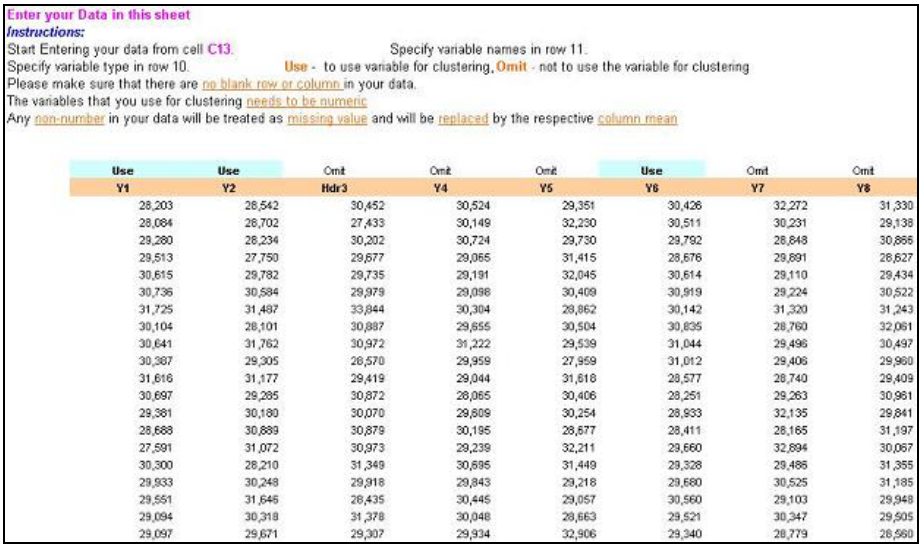
Hoja Input



En esta hoja se ingresar los parámetros de configuración del funcionamiento del algoritmo. Estos datos deben ser consistentes con los ingresados en la hoja "Data".
Se debe ingresar:

- Número de observaciones. (valor entre 5 y 5.000)
- Número de variables. (valor entre 3 y 50)
- Las dimensiones del mapa, valor que será elevado a la potencia 2 (dos) para obtener el total de neuronas del mapa. (entre 2 y 100)
- Número de ciclos de entrenamiento. (entre 1 y 1.000)
- Parámetros de aprendizaje (mayor a 0 y menor que 1)
    - Valor inicial
    - Valor final
    - Forma del decrecimiento = Exponencial o Lineal
- Valor de Sigma para la vecindad Gaussiana, como porcentaje del ancho del mapa.
    - Valor inicial
    - Valor final
    - Forma de decrecimiento = Exponencial o Lineal

Hoja Data



Enter your Data in this sheet

Instructions:

Start Entering your data from cell C13. Specify variable names in row 11.

Specify variable type in row 10. Use - to use variable for clustering, Omit - not to use the variable for clustering

Please make sure that there are no blank row or column in your data.

The variables that you use for clustering needs to be numeric

Any non-number in your data will be treated as missing value and will be replaced by the respective column mean

Use	Use	Omit	Omit	Omit	Use	Omit	Omit
Y1	Y2	Hdr3	Y4	Y5	Y6	Y7	Y8
28,203	28,542	30,452	30,524	29,351	30,426	32,272	31,330
28,084	28,702	27,433	30,149	32,230	30,511	30,231	29,138
29,280	28,234	30,202	30,724	29,730	29,792	28,848	30,866
29,513	27,750	29,677	29,065	31,415	28,676	29,891	28,627
30,615	29,782	29,735	29,191	32,045	30,614	29,110	29,434
30,736	30,584	29,979	29,098	30,409	30,919	29,224	30,522
31,725	31,487	33,844	30,304	28,862	30,142	31,320	31,243
30,104	28,101	30,887	29,855	30,504	30,835	28,760	32,061
30,641	31,762	30,972	31,222	29,539	31,044	29,496	30,497
30,387	29,305	28,570	29,959	27,959	31,012	29,406	29,960
31,616	31,177	29,419	29,044	31,618	28,577	28,740	29,409
30,697	29,285	30,872	28,065	30,406	28,251	29,263	30,961
29,381	30,180	30,070	29,809	30,254	28,933	32,135	29,841
28,688	30,889	30,879	30,195	28,677	28,411	28,165	31,197
27,591	31,072	30,973	29,239	32,211	29,660	32,894	30,067
30,300	28,210	31,349	30,695	31,449	29,328	29,486	31,355
29,933	30,248	29,918	29,843	29,218	29,680	30,525	31,185
29,551	31,646	28,435	30,445	29,057	30,560	29,103	29,948
29,094	30,318	31,378	30,048	28,663	29,521	30,347	29,505
29,097	29,671	29,307	29,934	32,906	29,340	28,779	28,560

Los datos deben ingresarse a partir de la celda C13

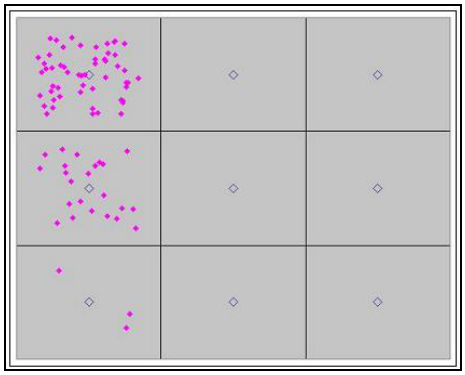
Los nombres de las variables deben ingresarse en la fila 11.

El tipo de variable debe especificarse en la fila 10 Para que la variable sea incluida por el algoritmo se debe indicar la opción "USE", para que sea ignorada "OMIT".

No debe haber filas o columnas con datos en blanco. Las variables que se usan para clasificar deben estar en formato numérico.

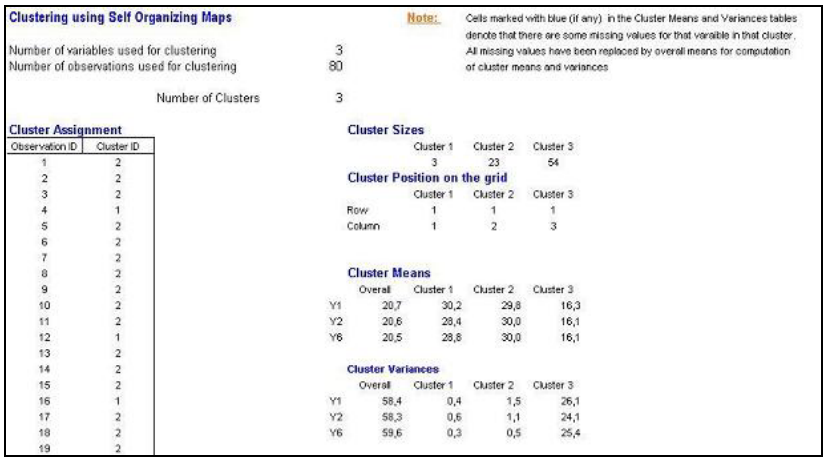
Los datos que no están en formato numérico serán considerados como datos perdidos y serán reemplazados por un valor de la misma columna.

Hoja Weihgts



Visualiza el mapa bidimensional. Las observaciones se van ubicando en cada cluster a medida que la aplicación se está ejecutando.

Hoja Output



Clustering using Self Organizing Maps

Number of variables used for clustering 3
Number of observations used for clustering 80

Number of Clusters 3

Note: Cells marked with blue (if any) in the Cluster Means and Variances tables denote that there are some missing values for that variable in that cluster. All missing values have been replaced by overall means for computation of cluster means and variances

Cluster Assignment

Observation ID	Cluster ID
1	2
2	2
3	2
4	1
5	2
6	2
7	2
8	2
9	2
10	2
11	2
12	1
13	2
14	2
15	2
16	1
17	2
18	2
19	2

Cluster Sizes

	Cluster 1	Cluster 2	Cluster 3
	3	23	54

Cluster Position on the grid

	Cluster 1	Cluster 2	Cluster 3
Row	1	1	1
Column	1	2	3

Cluster Means

	Overall	Cluster 1	Cluster 2	Cluster 3
Y1	20,7	30,2	29,8	16,3
Y2	20,6	28,4	30,0	16,1
Y6	20,5	28,8	30,0	16,1

Cluster Variances

	Overall	Cluster 1	Cluster 2	Cluster 3
Y1	58,4	0,4	1,5	26,1
Y2	58,3	0,6	1,1	24,1
Y6	59,6	0,3	0,5	25,4

En esta hoja presenta los resultados del algoritmo.
Informa:

- El número de variables usadas para clasificar
- El número de observaciones usadas para clasificar
- Cantidad de grupos (clusters)
- Tabla “Cluster Assignment” (Asignación de clusters): muestra por cada observación, (representada por una ID) el número de cluster asignado
- Tabla “Clusters Size” (Tamaño de los clusters): muestra la cantidad de observaciones encontradas en cada uno de los clusters o grupos.
- Tabla “Cluster Position on the grid” (Posición de cada cluster dentro de la grilla). Tabla de doble entrada donde se indica el número de fila y de columna que le corresponde a cada cluster.
- Tabla “Cluster Means” (Promedio de los clusters): Tabla de doble entrada donde se indica los valores promedio para la totalidad de los datos, y para cada uno de los clusters.
- Tabla “Cluster Variantes” (Varianza de los clusters): Tabla de doble entrada donde se indica la varianza para la totalidad de los datos y para cada uno de los clusters.

Hay una nota, que indica que aquellas celdas que estén de color azul en la tabla "Clustes Means" y "Clusters Variances", advierte la pérdida de valores para esa variable dentro del cluster. Todos los valores perdidos fueron reemplazados por valores medios o varianzas del cluster.

Hoja Junk y Hoja Plot

En esta hoja la aplicación guarda datos de soporte para generar los resultados

4.2.2 Bases de Datos de ejemplo

La siguiente base de datos [Solar Flare Databases, 1989] contiene registros de erupciones solares. Cada registro representa un conjunto de rasgos capturados en una región activa del sol ocurrida durante un lapso de 24 horas.

Información de los atributos:

- Código de clase - clase modificada por Zurich – (A, B, C, D, E, F, H)
- Código de medida del punto más grande (X, R, S, A, H, K)
- Código del espacio de distribución (X, O, I, C)
- Actividad (1 = reducida, 2 = sin alterar)
- Evolución (1 = decadente, 2 = no creciente, 3 = creciente)
- Código de actividad eruptiva de las 24 horas previas (1 = menor a un M1, 2 = M1, 3 = mayor a un M1)
- Históricamente complejo (1 = Sí, 2 = No)
- La región se volvió históricamente compleja con este episodio (1 = Sí, 2 = No)
- Área (1 = chica, 2 = grande)
- Área del punto más grande (1 = <=5, 2 = >5)

Código de Clase	Código de medida del punto mas grande	Código del espacio de distribución	Actividad	Evolución	Código de actividad eruptiva de las 24 horas previas	Históricamente complejo	La región se volvió históricamente compleja con este episodio	Área	Área del punto mas grande
C	S	O	1	2	1	1	2	1	2
D	S	O	1	3	1	1	2	1	2
C	S	O	1	3	1	1	2	1	1
D	S	O	1	3	1	1	2	1	2
D	A	O	1	3	1	1	2	1	2
D	A	O	1	2	1	1	2	1	2
D	A	O	1	2	1	1	2	1	1
D	A	O	1	2	1	1	2	1	2
D	K	O	1	3	1	1	2	1	2
C	R	O	1	3	1	1	2	1	1

Se deben convertir los campos “Código de clase”, “Código de medida del punto más grande” y “Código del espacio de distribución” en campos de tipo numéricos, dado que no podrán ser utilizados por la aplicación.

Lo que se pretende es que el aplicativo del algoritmo SOM, en base a los datos ingresados, clasifique estos registros de manera tal que se puedan obtener grupos con características similares.

4.3 Herramienta de aplicación de C4.5 (CTree)

[CTree, 1998]

4.3.1 Funcionamiento de la herramienta

Hoja ReadMe

Presenta una descripción en inglés de las instrucciones para el uso de la herramienta.

Paso 1: Ingresar los datos

(A) Se deben ingresar los datos en la hoja Data, empezando por la celda L24. Se pueden ingresar un total de filas entre 10 y 10.000.
(B) Las observaciones deben ubicarse en filas y las variables en columnas.
(C) Debe elegirse en cada columna el Tipo apropiado (Omit, Class, Cont, Cat)

- Si se quiere excluir la columna se debe seleccionar: Omit
- Para que la columna funcione como categoría de predicción se debe seleccionar: Cat
- Para que la columna funcione como predicción continua, se debe seleccionar: Cont
- Para que la columna funcione como variable de clase, se debe seleccionar: Class

Se puede tener un máximo de 50 variables. Debe haber sólo una clase, veinte como máximo de tipo Cat, incluida la de tipo Class.
(D) No deben haber filas o columnas en blanco.
(E) La variable de tipo Class, no puede contener valores nulos.
(F) Cualquier dato de tipo no numérico en una columna de tipo Cont, será considerado como un valor perdido; y la aplicación lo reemplazará por la media de la columna.
(G) Cualquier celda en blanco o con error de Excel, en una columna de tipo Cat, será considerado como un valor perdido; la aplicación lo reemplazará por el valor de mayor frecuencia de ocurrencia en la misma columna.
La aplicación no diferencia en los nombres de las columnas entre, por ejemplo; good, Good, GOOD, etc. Todos serán tratados como misma categoría.
Debe haber como mínimo dos observaciones por cada columna tipo Cat. Si hay sólo una se debe, o bien eliminar la observación o renombrar la categoría hacia otra de la misma columna.

Paso 2: Configurar el modelo

(A) Se deben completar los datos de la hoja “UserInput”.
(B) Se debe tener cuidado que los valores estén dentro de los rangos aceptados por la aplicación.
(C) Presionar el botón “Build Tree” para comenzar el modelo.

Paso 3: Resultados

(A) Al finalizar se puede observar el árbol de clasificación en la hoja “Tree”. En esta hoja se ingresan lo valores de predicción y en la celda H7 se observa la clase de predicción generada por el árbol.
(B) Se puede seleccionar una celda en cualquier de los nodos y chiquear el botón “View Node” para ver los detalles de información de este nodo en la hoja “NodeView”.

(C) En la celda F7 de la hoja "NodeView", se puede ingresar cualquier número de nodo para ver la clase de distribución y alguna otra información acerca del nodo.

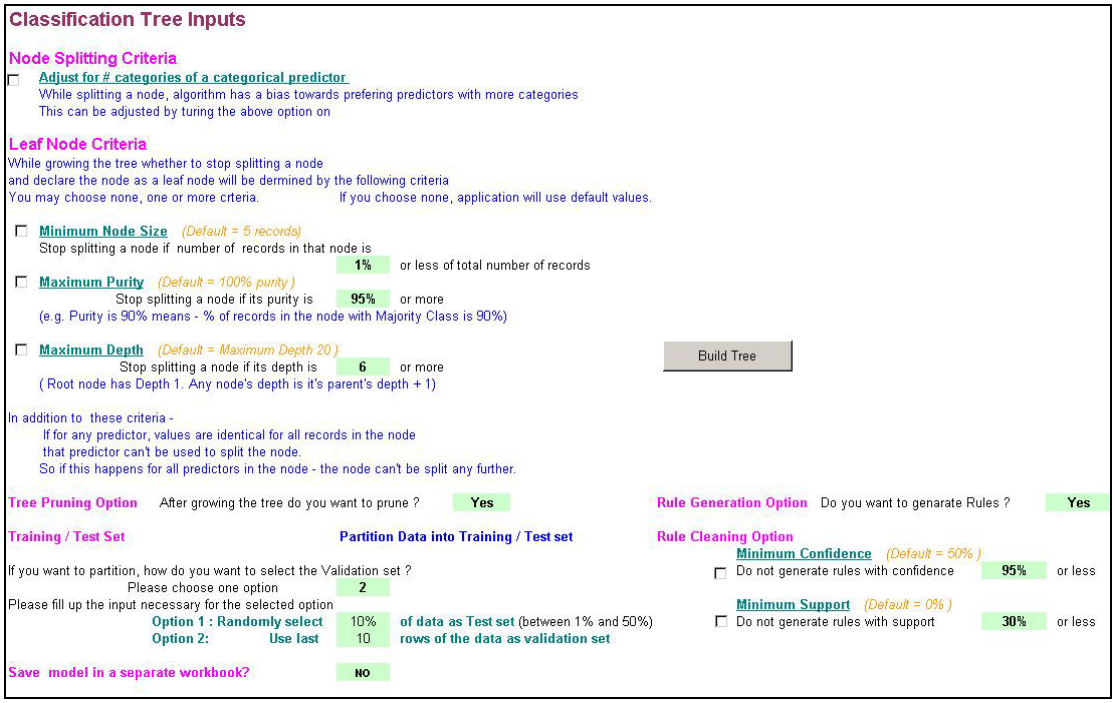
Paso 4: Generación de reglas.

(A) Las reglas se generan luego de que el árbol se desarrolló. La aplicación sólo genera reglas, no está preparada para que estas reglas clasifiquen nuevos datos. obtener ninguna información a partir de ellas. La tabla que contiene el resumen de las reglas informa acerca de la calidad individual de cada una. La calidad se mide según tres métricas que a continuación se explican. Por ejemplo, considérese la regla:
"SI ancho de pétalo > 7 Y largo de pétalo <= 70 ENTONCES especie = setosa"
A continuación se explica la calidad de la regla arriba descripta:
SOPORTE: es el porcentaje de datos de entrenamiento que justifican como verdadera la parte izquierda de la regla. Si para una determinada observación, la parte izquierda de la regla es verdadera, decimos que esta es la regla aplicable a dicha observación. La medida está dada por la cantidad de observaciones sobre las que se aplica la regla.
CONFIDENCIALIDAD: fuera de los registros de entrenamiento para los cuales la parte izquierda de la regla es verdadera, existe un porcentaje de registros para los cuales la parte derecha de la regla también es verdadera. En otras palabras, representa el porcentaje de observaciones para la cual la regla es aplicable, es decir para la cual la regla es verdadera. La medida está dada por el alcance de la regla.
REPRESENTATIVIDAD: es el porcentaje de registros de setosa que fueron correctamente capturados por la regla. Esto lleva a una reflexión acerca de la estructura del problema. Si existe una regla que captura el 100% del total, esto significa que en el espacio de predicción, todas las observaciones de esta clase cierran entre sí y que la regla está habilitada para captar correctamente la parte el espacio de predicción.
Suponga que hay 150 observaciones en los datos de entrenamiento, de los cuales 50 pertenecen a la clase de la setosa. Además se sabe que la regla arriba descripta se aplica sobre 70 registros, de los cuales 30 son de setosa.
Los resultados son los siguientes:

SOPORTE: 47% (= 70 / 150)
CONFIDENCIALIDAD: 43% (= 30 / 70)
REPRESENTATIVIDAD: 60 % (= 30 / 50)

Algunos puntos más a tener en cuenta
(A) Ajuste de diferentes categorías para el predictor
Mientras se va creando el árbol, los nodos hijos son creados por la partición de los nodos padres. El predictor a usar en esta partición es una decisión que requiere cierto criterio. El criterio tiene cierta propensión a elegir predictores con más categorías. Esta tendencia puede ser ajustada seleccionando esta opción.
(B) Criterio de tamaño del nodo mínimo
Esto no conviene seleccionarse. En el caso de hacerlo se debe ingresar un tamaño de nodo mínimo que sea válido, expresado como porcentaje del total de las observaciones. Un valor de tamaño mínimo del nodo debería ser exactamente mayor al 0% y exactamente menor a 100%. El aumentar este valor, genera un árbol más pequeño.
(C) Criterio máximo de purificación

Hoja UserInput



Entrada para el árbol de clasificación

Criterios para partición de nodos.

Ajuste # categorías para un predictor de categorías

Cuando el nodo se divide, el algoritmo tiende a preferir predictores con más categorías. Esto puede ser activado, indicado el estado ON en la casilla correspondiente.

Criterios de ramificación

A medida que el árbol se va desarrollando, termine o no de ramificarse un nodo y se declare al nodo como un nodo hoja, puede ser determinado por los siguientes criterios. Se puede no elegir ningún criterio, uno o varios. Si no se elige ningún criterio, la aplicación usa los valores por defecto.

- Tamaño mínimo del nodo (Valor por defecto = 5 registros)
  El nodo no se ramifica más si el número de registros en el nodo es = (porcentaje a ingresar) o menor al número total de registros.
- Nivel máximo de pureza (Valor por defecto = 100% de pureza)
  El nodo no se ramifica más si el valor de pureza es = (porcentaje a ingresar) o mayor. (Por ejemplo si la pureza es del 90 % significa que ese porcentaje de registros en el nodo pertenecen en un 90 % a la clase mayoritaria)
- Nivel máximo de profundidad (Valor por defecto = 20 es el máximo nivel de profundidad)
  El nodo no se ramifica más si el valor de la profundidad es = (valor a ingresar) o mayor. (El nodo raíz tiene profundidad 1. Cualquier nodo dependiente es igual a la profundidad de su nodo padre + 1).

A estos criterios cabe agregar:

Si para algún predictor, los valores son idénticos para todos los registros del nodo, entonces ese predictor puede ser usado para ramificar el nodo. Aunque si esto sucede para todos los predictores del nodo, este nodo no podrá de ningún modo ser ramificado.

Opciones de poda del árbol.

Luego que el árbol se ha desarrollado, se puede seleccionar la posibilidad de realizar una poda (Si o No).

Entrenamiento / Configuración de prueba

Se debe seleccionar si:

Se usan todos los datos para el entrenamiento

Se usa una parte de los datos

En el caso que se opte por usar una parte de los datos, se debe indicar la forma de seleccionar la configuración de la validación. Se puede elegir la opción 1 o la 2.

- La opción 1: Selecciona de manera aleatoria un porcentaje (valor a ingresar entre 1% y 50%) de datos como datos de prueba.
- La opción 2: Usa las últimas (valor a ingresar) filas de datos como datos de validación

Guardar el modelo en una hoja separada? (Ingresar Si o No)

Opciones para generación de reglas. Ingresar si se desea generar reglas (Si o No)

Opciones para limpieza de reglas.

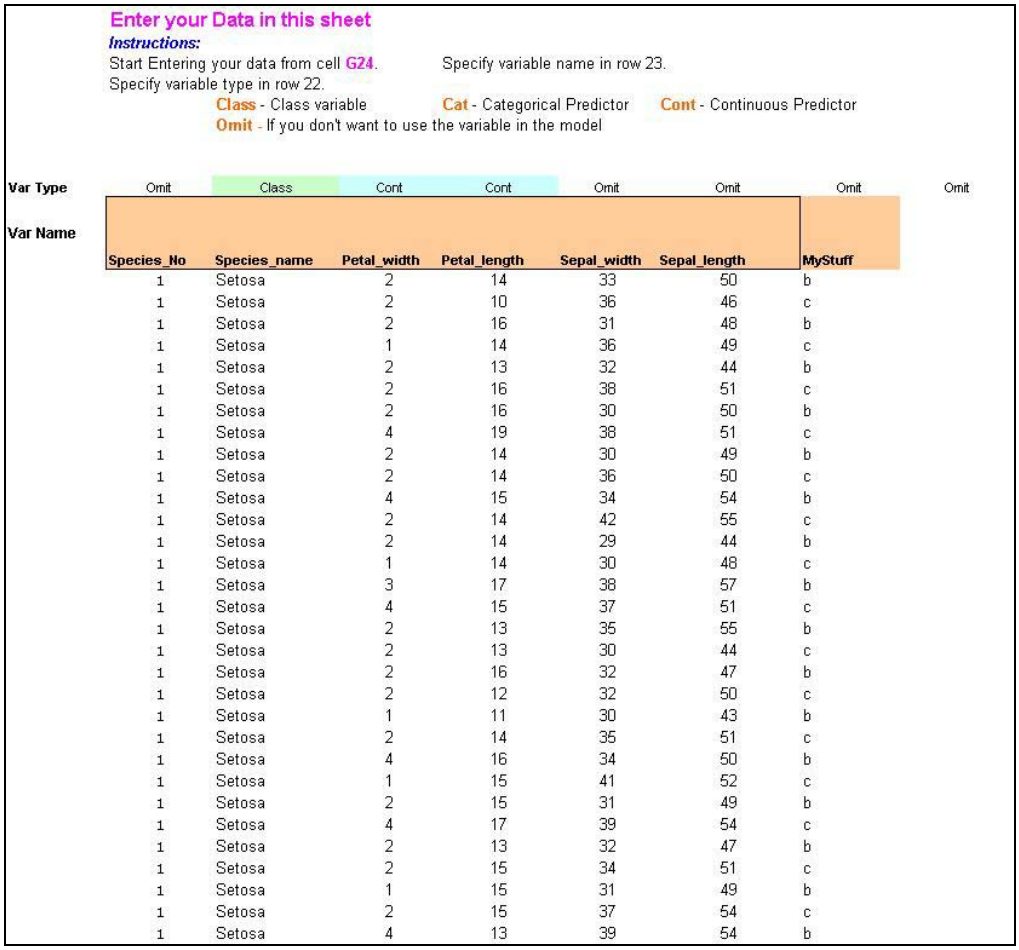
Mínima confidencia (Valor por defecto = 50 %)

No se genera reglas con confidencia = (porcentaje a ingresar) o menor.

Máximo soporte (Valor por defecto = 0 %)

No se genera reglas con soporte = (porcentaje a ingresar) o menor.

Hoja Data



Enter your Data in this sheet

Instructions:

Start Entering your data from cell G24. Specify variable name in row 23.

Specify variable type in row 22.

Class - Class variable Cat - Categorical Predictor Cont - Continuous Predictor

Omit - If you don't want to use the variable in the model

Var Type	Omit	Class	Cont	Cont	Omit	Omit	Omit	Omit
Var Name	Species_No	Species_name	Petal_width	Petal_length	Sepal_width	Sepal_length	MyStuff	
	1	Setosa	2	14	33	50	b	
	1	Setosa	2	10	36	46	c	
	1	Setosa	2	16	31	48	b	
	1	Setosa	1	14	36	49	c	
	1	Setosa	2	13	32	44	b	
	1	Setosa	2	16	38	51	c	
	1	Setosa	2	16	30	50	b	
	1	Setosa	4	19	38	51	c	
	1	Setosa	2	14	30	49	b	
	1	Setosa	2	14	36	50	c	
	1	Setosa	4	15	34	54	b	
	1	Setosa	2	14	42	55	c	
	1	Setosa	2	14	29	44	b	
	1	Setosa	1	14	30	48	c	
	1	Setosa	3	17	38	57	b	
	1	Setosa	4	15	37	51	c	
	1	Setosa	2	13	35	55	b	
	1	Setosa	2	13	30	44	c	
	1	Setosa	2	16	32	47	b	
	1	Setosa	2	12	32	50	c	
	1	Setosa	1	11	30	43	b	
	1	Setosa	2	14	35	51	c	
	1	Setosa	4	16	34	50	b	
	1	Setosa	1	15	41	52	c	
	1	Setosa	2	15	31	49	b	
	1	Setosa	4	17	39	54	c	
	1	Setosa	2	13	32	47	b	
	1	Setosa	2	15	34	51	c	
	1	Setosa	1	15	31	49	b	
	1	Setosa	2	15	37	54	c	
	1	Setosa	4	13	39	54	b	

En esta hoja deben ingresar los datos a procesar.

Los datos deben comenzar a ingresarse a partir de la celda G24. Los nombres de variables se deben ingresar en la fila 23. Los tipos de variables se especifican en la fila 22.

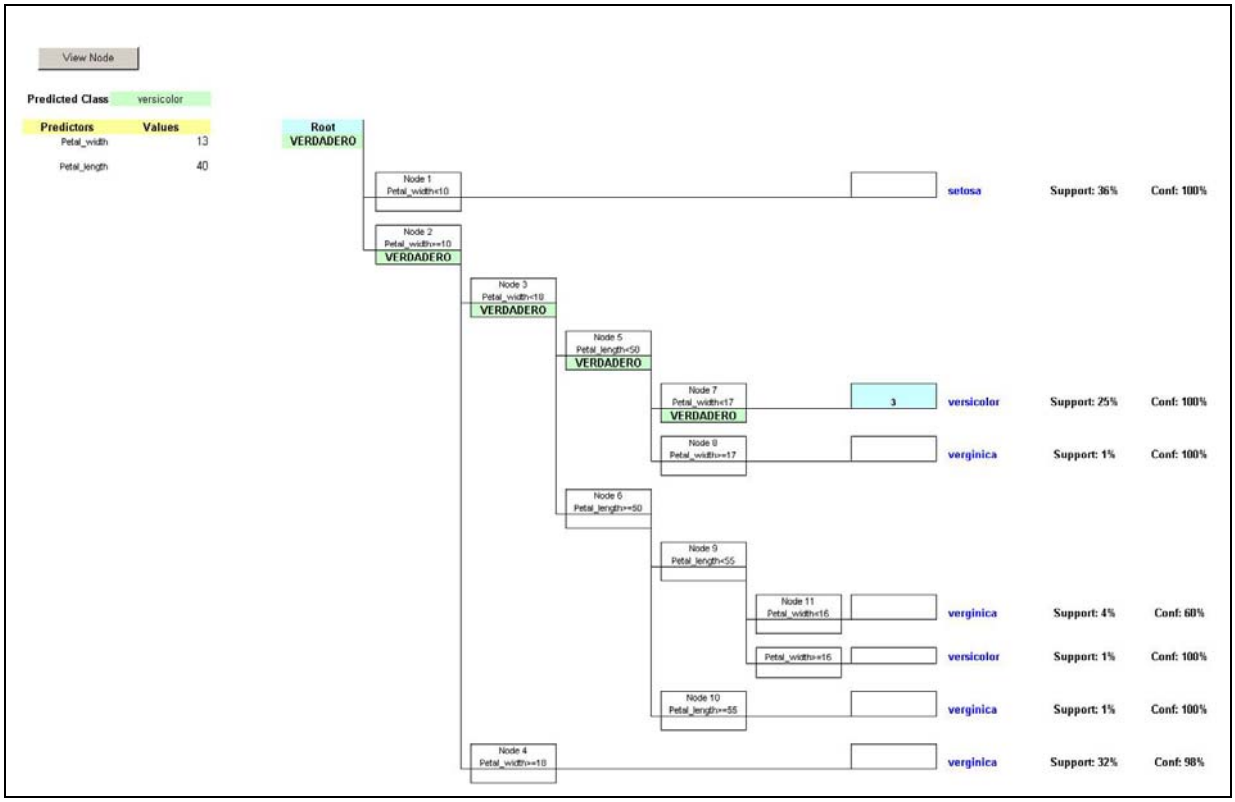
Class: variable de clase

Cat: atributo de tipo categórico

Cont: atributos de tipo continuo

Omit: cuando se quiere excluir la variable del modelo.

Hoja Tree

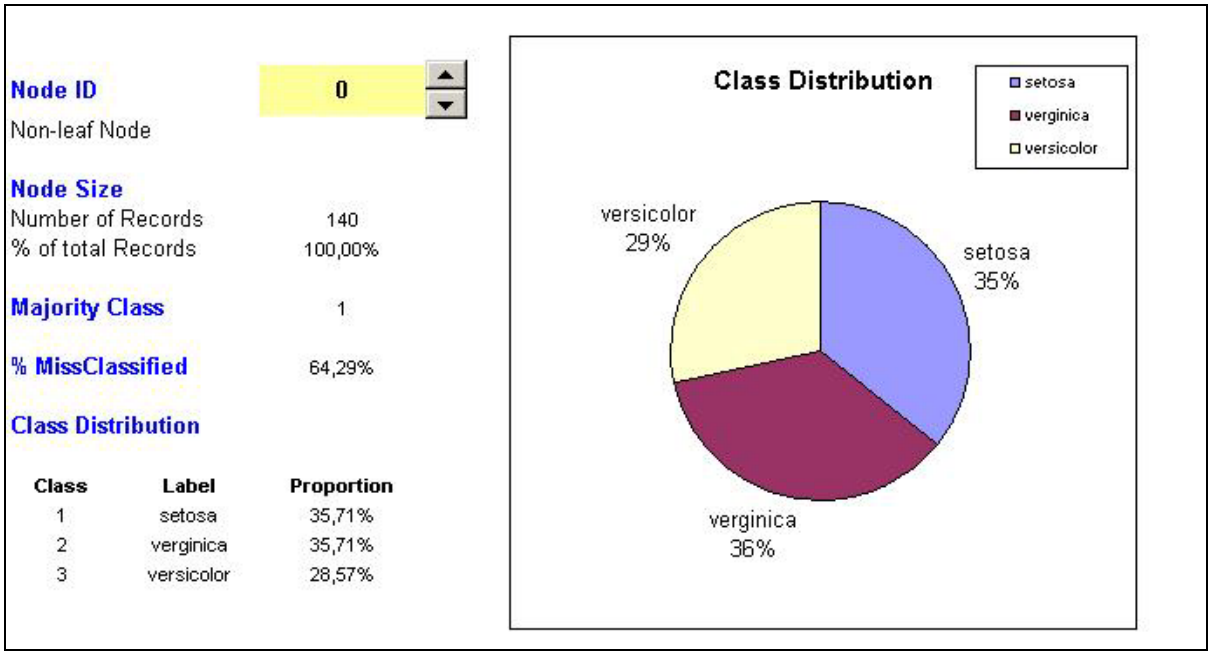


En esta hoja se visualiza el árbol que generó el modelo. Si se selecciona alguno de los nodos, se puede ver información de cada nodo en la hoja "NodeView", presionando el botón "View Node"

Visualiza el resultado de la clase predictora.

Visualiza una tabla con los nombres de los predictores y los valores de cada uno.

Hoja NodeView



Esta hoja muestra información de cada nodo seleccionado en la hoja anterior, presionando el botón "View Node".

Muestra por cada nodo, un gráfico de tortas con los porcentajes obtenidos en la clase.

Se indica el ID del nodo.

El tamaño del nodo

Número de registros en el nodo

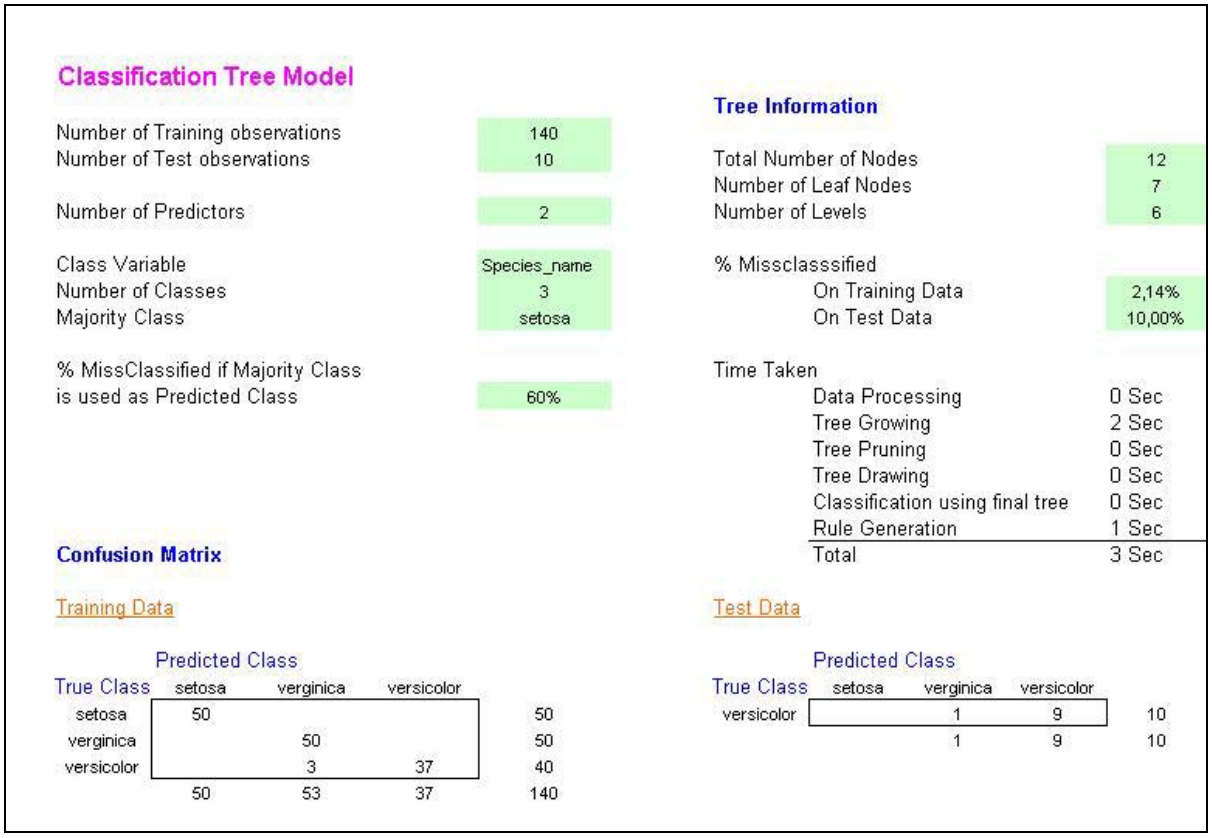
Porcentaje total de registros encontrados en el nodo

La clase mayoritaria (la de porcentaje mayor)

El porcentaje errado (la suma de las demás clases perdedoras)

La distribución de clases. El número de clase, la denominación y el porcentaje obtenido en el nodo seleccionado

Hoja Result



Classification Tree Model

Number of Training observations	140
Number of Test observations	10
Number of Predictors	2
Class Variable	Species_name
Number of Classes	3
Majority Class	setosa
% MissClassified if Majority Class is used as Predicted Class	60%

Tree Information

Total Number of Nodes	12
Number of Leaf Nodes	7
Number of Levels	6
% Missclasssified	
On Training Data	2,14%
On Test Data	10,00%
Time Taken	
Data Processing	0 Sec
Tree Growing	2 Sec
Tree Pruning	0 Sec
Tree Drawing	0 Sec
Classification using final tree	0 Sec
Rule Generation	1 Sec
Total	3 Sec

Confusion Matrix

Training Data

True Class \ Predicted Class	setosa	verginica	versicolor	
setosa	50			50
verginica		50		50
versicolor		3	37	40
	50	53	37	140

Test Data

True Class \ Predicted Class	setosa	verginica	versicolor	
versicolor		1	9	10
		1	9	10

Se visualizan los resultados de árbol que generó el modelo.

- Número de observaciones para el entrenamiento
- Número de observaciones de prueba
- Número de predictores.
- Nombre de la clase variable.
- Número de clases
- Clase mayoritaria
- Porcentaje no clasificado, cuando la clase mayoritaria es usada como clase predictiva.

Información del árbol

- Número total de nodos.
- Número total de nodos hoja.
- Cantidad de niveles
- Porcentaje no clasificado
  - *En los datos de entrenamiento*
  - *En los datos de prueba*
- Tiempo utilizado
  - *En el procesamiento de datos*
  - *En el desarrollo del árbol*
  - *En la poda del árbol*
  - *En el diseño del árbol*
  - *En la clasificación utilizando el árbol final*
  - *En la generación de reglas.*
  - *Tiempo Total*

Matriz de confusión

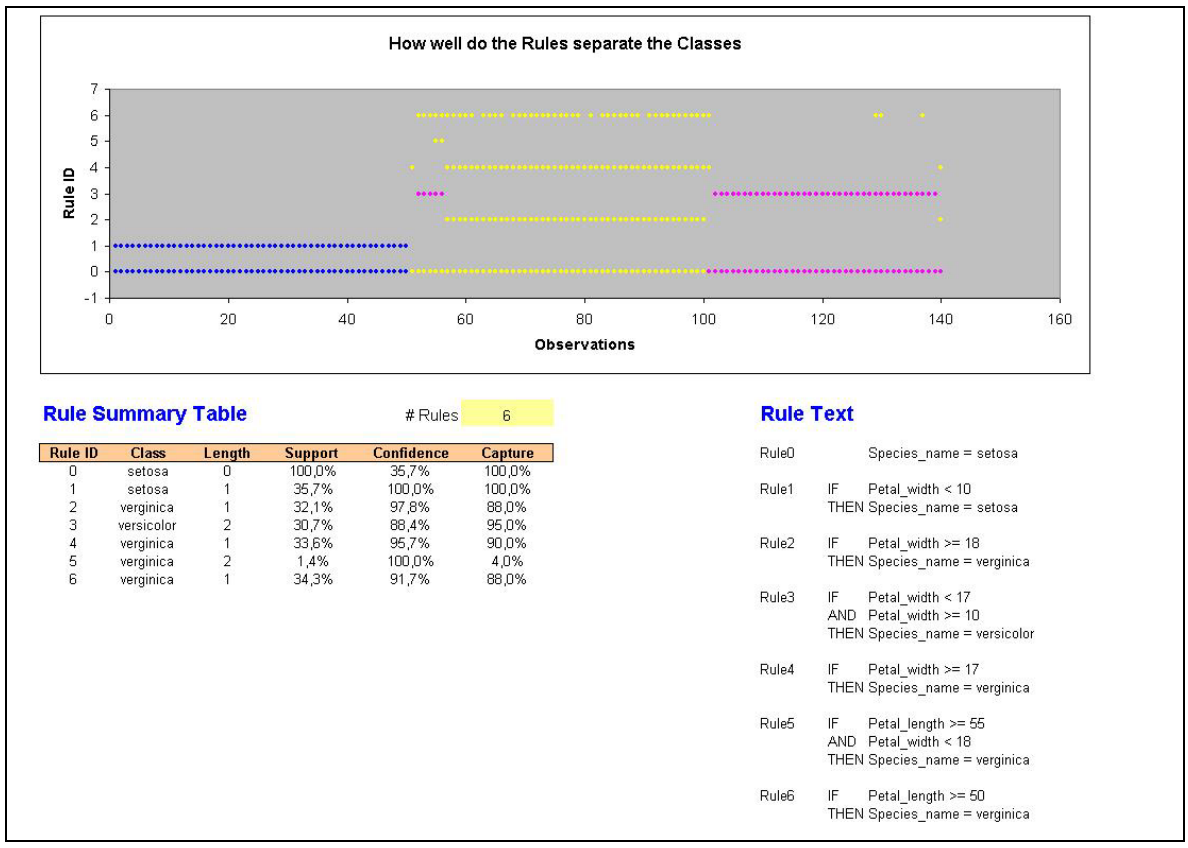
Datos de entrenamiento

Tabla de doble entrada con los resultados obtenidos en el entrenamiento a partir de las clases predictivas.

Datos de prueba

Tabla de doble entrada con los resultados obtenidos a partir de las clases predictivas en la prueba

Hoja Rules



Rule Summary Table # Rules 6

Rule ID	Class	Length	Support	Confidence	Capture
0	setosa	0	100,0%	35,7%	100,0%
1	setosa	1	35,7%	100,0%	100,0%
2	verginica	1	32,1%	97,8%	88,0%
3	versicolor	2	30,7%	88,4%	95,0%
4	verginica	1	33,6%	95,7%	90,0%
5	verginica	2	1,4%	100,0%	4,0%
6	verginica	1	34,3%	91,7%	88,0%

Rule Text

Rule0 Species_name = setosa

Rule1 IF Petal_width < 10
THEN Species_name = setosa

Rule2 IF Petal_width >= 18
THEN Species_name = verginica

Rule3 IF Petal_width < 17
AND Petal_width >= 10
THEN Species_name = versicolor

Rule4 IF Petal_width >= 17
THEN Species_name = verginica

Rule5 IF Petal_length >= 55
AND Petal_width < 18
THEN Species_name = verginica

Rule6 IF Petal_length >= 50
THEN Species_name = verginica

En esta hoja se escriben las reglas generadas por el modelo.

En el gráfico se observa la cantidad de observaciones obtenidas para cada clase con cada una de las reglas generada. (El color indica la clase, en el eje de las X están las reglas y en el eje de las Y están las observaciones.

La tabla con los resultados de las reglas, indica la cantidad total de reglas obtenidas.

Por cada una de ellas indica, su ID, la clase resultante, la longitud, el porcentaje de soporte, el porcentaje de confidencialidad y el porcentaje de captura.

También se transcribe el texto de cada una de las reglas.

4.3.2 Bases de Datos de ejemplo

La base de datos que se presenta como ejemplo de aplicación del algoritmo C4.5 para ser aplicada con la aplicación CTree, representa los votos obtenidos de los congresistas en Estados Unidos, en el año 1985. Se realizaron votaciones por temas diversos, a los datos obtenidos se les agregó un campo donde indica la clase a la que pertenece el funcionario; a saber: "Republicanos" y "Federales". Se identificaron tres tipos diferentes de votos: "A favor", "En contra", o "Desconocido", para cada ítem. Se pretende que el algoritmo construya las reglas de inducción para determinar cierto tipo de pensamiento entre uno y otro tipo de pensamiento político. [Servente et al, 2002]

Niños discapacitados	Participación en el costo del proyecto del agua	Adopción de la resolución sobre el presupuesto	Congelamiento de los honorarios médicos	Ayuda a El Salvador	Grupos religiosos en las escuelas	Prohibición de las pruebas anti-satélites	Ayuda a los contras de Nicaragua	Misil mx	Inmigración	Reducción a la corporación Synfuels	Presupuesto de educación	Derecho a demandar de la Superfund	Crimen	Exportaciones sin impuestos	Acta sudafricana de administración de exportaciones	Clase
A favor	En contra	A favor	En contra	A favor	A favor	En contra	En contra	En contra	En contra	En contra	En contra	En contra	En contra	En contra	A favor	Demócrata
En contra	A favor	En contra	A favor	A favor	A favor	En contra	En contra	En contra	En contra	A favor	A favor	A favor	A favor	En contra	En contra	Republicano
A favor	A favor	A favor	En contra	A favor	A favor	En contra	A favor	En contra	En contra	A favor	En contra	A favor	En contra	A favor	A favor	Demócrata
En contra	A favor	A favor	En contra	En contra	En contra	A favor	A favor	A favor	A favor	A favor	En contra	En contra	En contra	A favor	A favor	Demócrata
En contra	En contra	En contra	En contra	A favor	A favor	A favor	En contra	En contra	En contra	En contra	A favor	A favor	A favor	En contra	A favor	Demócrata
A favor	A favor	A favor	En contra	En contra	En contra	A favor	A favor	A favor	A favor	A favor	En contra	En contra	En contra	En contra	A favor	Demócrata
A favor	A favor	A favor	En contra	En contra	En contra	A favor	A favor	A favor	A favor	En contra	En contra	A favor	En contra	En contra	A favor	Demócrata
En contra	desconocido	A favor	En contra	En contra	En contra	A favor	A favor	A favor	A favor	A favor	En contra	En contra	A favor	A favor	A favor	Demócrata
A favor	A favor	A favor	En contra	En contra	En contra	En contra	A favor	A favor	En contra	A favor	En contra	En contra	En contra	A favor	A favor	Demócrata
En contra	A favor	A favor	En contra	En contra	A favor	A favor	A favor	A favor	En contra	desconocido	En contra	En contra	En contra	En contra	A favor	Demócrata
A favor	En contra	A favor	En contra	En contra	A favor	A favor	A favor	A favor	A favor	A favor	En contra	En contra	En contra	En contra	A favor	Demócrata
En contra	A favor	En contra	A favor	A favor	A favor	En contra	En contra	En contra	En contra	En contra	A favor	A favor	A favor	En contra	En contra	Republicano

5- CONCLUSIÓN

La Minería de Datos está comenzando a cobrar mayor peso como consecuencia de la gran masa de datos de la cual hoy las empresas u organizaciones son propietarias. Esto es una evolución de los datos hacia una nueva dimensión, los sistemas están empezando a demostrar que no sólo sirven para generar muy buenos reportes, eficientes automatizaciones, seguros almacenes de datos. La tecnologías de minería de datos, junto con las redes neuronales y el aprendizaje automático, muestran tener habilidades, todavía algo desdibujadas, de lo necesarias que son a la hora de resolver situaciones que se encuentran fuera del alcance de los sistemas de base de datos convencionales.

El camino hacia un resultado exitoso, todavía está por hacerse. No basta con algoritmos, que son pieza fundamental en esta ardua tarea; hacen falta herramientas que automaticen los pasos previos a la exploración de datos como así también aplicaciones que puedan refinar las salidas que resultas de estos procesos.

6- BIBLIOGRAFÍA

[García Martínez et al, 2003] García Martínez, R.; Servente, M.; Pasquín, D.; 2003. ***Sistemas Inteligentes,*** Capítulo 1: "Aprendizaje Automático", Capítulo 2 "Redes Neuronales Artificiales"; Nueva Librería, Buenos Aires, Argentina.

[Nnclass, 1998] Angshuman Saha; Tesis Doctoral ***Application of Ridge Regression for Improved Estimation of Parameters in Compartmental Models***; Departamento de Estadística; Universidad de Washingtong; Agosto 1998 http://www.geocities.com/adotsaha/NN/SOMinExcel.html. Agosto 1998.

[Nnclust, 1998] Angshuman Saha; Tesis Doctoral ***Application of Ridge Regression for Improved Estimation of Parameters in Compartmental Models***; Departamento de Estadística; Universidad de Washintong; Agosto 1998 http://www.geocities.com/adotsaha/NN/SOMinExcel.html. Agosto 1998.

[CTree, 1998] Angshuman Saha; Tesis Doctoral ***Application of Ridge Regression for Improved Estimation of Parameters in Compartmental Models;*** Departamento de estadística; Universidad de Washintong; Agosto 1998 http://www.geocities.com/adotsaha/CTree/CtreeinExcel.html

[Contraceptive Method Choice; 1987] National Indonesia Contraceptive Prevalence Survey; Contraceptive Method Choice; 1987. http://www.ics.uci.edu/~mlean/MLSummary.html

[Solar Flare Databases, 1989] Gary Bradshaw; Solar Flare Databases; 1989. http://www.ics.uci.edu/~mlean/MLSummary.html

[Redes Competitivas, 2000] Universidad Tecnológica de Pereira, Colombia, 2000 http://ohm.utp.edu.co/neuronales

[Servente et al, 2002] Servente, M; Dr. García Martínez, R; Tesis Doctoral ***Algoritmos TDIDT aplicado a la minería de datos inteligentes*** , Universidad de Buenos Aires, 2002