

Laboratorio de Investigación con **Clementine**



Seminario
“Data Mining”

Máster en Ingeniería del Software del DSIC

José Hernández Orallo. (jorallo@dsic.upv.es). Noviembre 2002

Objetivos:

Los objetivos principales de este laboratorio son:

- Aprender a manejar la herramienta de minería de datos SPSS Clementine: obtener y transformar datos, generar modelos y evaluarlos.
- Aplicarla para distintos tipos de problemas de minería de datos: descriptivos y predictivos.

El tiempo estimado para realizar este boletín es de cuatro horas (sin incluir los ejercicios libres).

Índice

1.	Introducción al SPSS Clementine	3
1.1	Sources (Orígenes)	4
1.2	Record Ops. (Oper. con registros).....	5
1.3	Field Ops. (Oper. con campos).....	5
1.4	Graph (Gráficos)	5
1.5	Modelling (Modelado)	6
1.6	Output (Salida).....	7
2.	Un primer ejemplo	8
3.	Visualización y Preparación de Datos.....	15
3.1	Enunciado de un Primer Problema. Selección de Fármaco	15
3.2	Resolución del Problema de Selección de Fármaco	15
3.3	Un Segundo Problema: Agrupación de Empleados	23
3.4	Resolución del Problema de Agrupación de Empleados	23
3.5	Obtención y Transformación de Datos Relacionales	26
4.	Validación de Modelos	36
4.1.1	Cross-Validation	37
4.1.2	Sampling & Boosting	39
5.	Patrones Predictivos.....	43
5.1	Patrones Secuenciales	43
5.2	Clasificación.....	44
5.2.1	Varios modelos para los medicamentos	44
5.2.2	Varios modelos para el monks1	44
6.	Patrones Descriptivos.....	44
6.1	Reglas de Asociación y Determinación.....	44
6.2	Estudios de Correlación	46
6.3	Segmentación	46
7.	Generación y Exportación de Resultados.....	46
8.	Combinación y Ponderación de Modelos.....	46
9.	Ejercicios Libres	47
9.1	Tamaño Pequeño/Medio:	47
9.2	Gran Tamaño.....	47
10.	Soluciones	48
10.1	Problema de la Cámara:	48
10.2	Problema del muestreo de fuentes de drugs (1700 – 500):.....	49
10.2.1	Muestreo no aleatorio:.....	49
10.2.2	Muestreo aleatorio:.....	49
10.3	Problema del Monks1:	50
10.4	Problema de la Cesta de la Compra:	51
10.5	Problema de segmentación de pacientes:	51

PRIMERA PARTE

En esta primera parte se van descubriendo paso a paso las posibilidades del Clementine.

1. Introducción al SPSS Clementine

El SPSS Clementine es una herramienta integrada de minería de datos, inicialmente de *Integral Solutions Limited* (ISL) y ahora de SPSS (www.spss.com).

La versión 5.2.1 en inglés o la 6.0.2 en castellano, que es con las que vamos a trabajar indistintamente¹, incluye las siguientes características:

- Diversas fuentes de datos (ASCII, XLS u ODBC).
- Interfaz visual basado en procesos/flujo de datos (streams).
- Distintas herramientas de minería de datos: correlación, reglas de asociación (GRI, a priori), patrones secuenciales (regresión), segmentación (Kohonen, Two-step y k-means), clasificación (redes neuronales, reglas y árboles de decisión).
- Manipulación de datos (pick & mix, muestreo, combinación y separación).
- Combinación de modelos.
- Visualización anterior (datos).
- Exportación de modelos a distintos lenguajes (C, SPSS, SAS).
- Exportación de datos integrada a otros programas (XLS).
- Generación de informes.

El entorno del Clementine está basado en *nodos* que se van disponiendo y conectando para formar un flujo (*stream*), traducido por Clementine también como “ruta”. Los streams pueden dejarse en ficheros separados (.str) o se pueden organizar en proyectos (.cpj). De hecho, tanto los streams como los proyectos de minería de datos se almacenan en ficheros separados que se puede abrir, modificar, reejecutar o reorganizar, y que son independientes de las fuentes de datos.

La Figura 1 muestra un ejemplo de stream con cinco nodos interconectados:

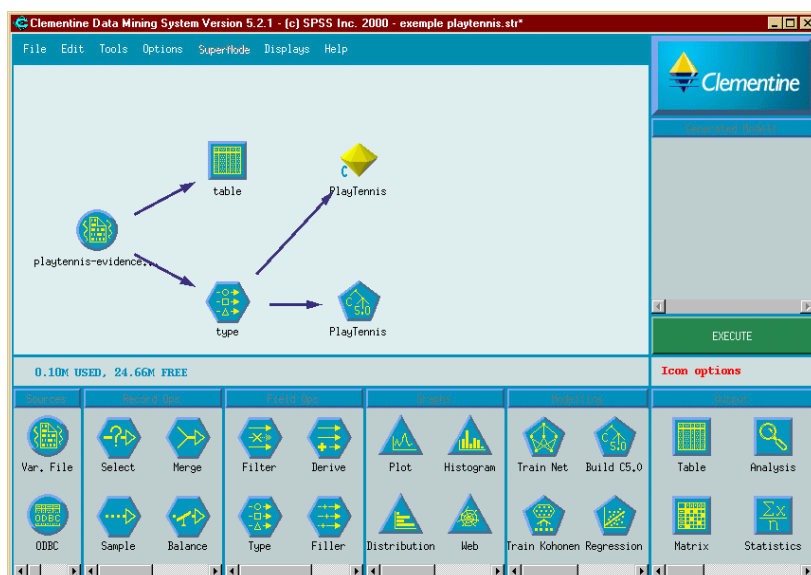


Figura 1. Un stream en Clementine.

¹ Debido a lo confuso de la traducción en castellano de la versión 6.0.2 y que además la ayuda sólo está en inglés, en lo sucesivo utilizaremos preferentemente la nomenclatura en inglés, indicando, inicialmente, también la traducción en castellano realizada por el Clementine 6.0.2.

Como se puede ver en la parte inferior de la Figura 1, el Clementine clasifica los nodos en seis categorías:

- Sources (Orígenes): nodos para obtener los datos de trabajo (fuentes de datos).
- Record Ops (Oper. con registros): operadores para modificar o combinar registros (filas) de distintas fuentes. Es decir, selecciones y combinaciones.
- Field Ops (Oper. con campos): operadores para modificar o combinar campos (columnas).
- Graphs (Gráficos): gráficas.
- Modelling (Modelado): tipos de modelos/patrones que puede generar Clementine
- Output (Salida): presentación de tablas, análisis de modelos, estadísticas, exportación de datos.

Los nodos disponibles en la versión 5.2.1 se muestran en la Figura 2:

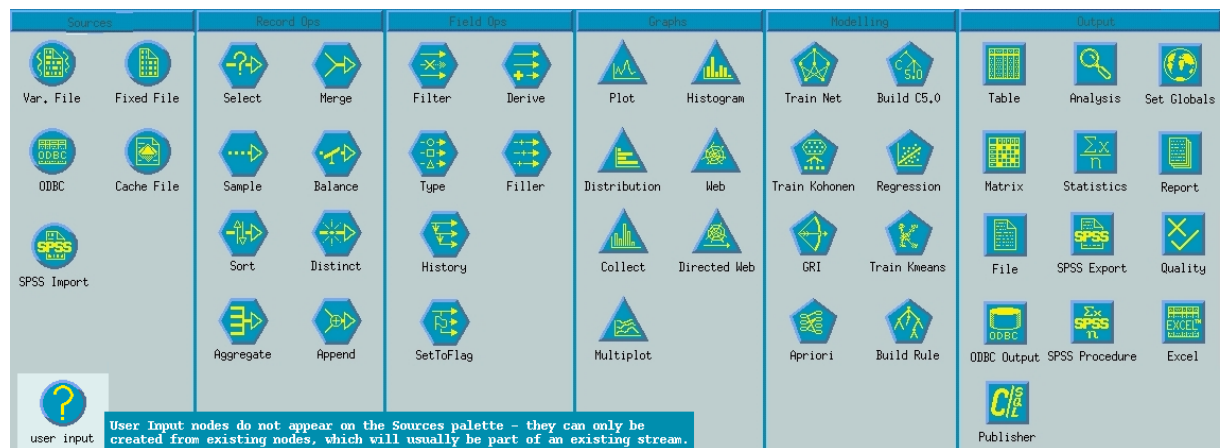


Figura 2. Nodos disponibles en Clementine 5.2.1

No existen variaciones importantes entre la versión 5.2.1 y 6.0.2 respecto a nodos disponibles. Se clarificará dicha diferencia en la siguiente sección. En cualquier caso, este boletín se puede desarrollar indistintamente en ambas versiones.

Pasemos a describir los nodos más importantes de las categorías anteriores:

1.1 Sources (Orígenes)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Var. File (Archivo variable)	✓	✓	Permiten leer datos de ficheros de texto que tengan un tamaño de caracteres variable por registro pero un número fijo de campos.
Fixed File (Archivo Fijo)	✓	✓	Permiten leer datos de ficheros de texto que tengan un tamaño fijo de caracteres por campo, con lo que su uso se limita a fuentes perfectamente tabuladas.
ODBC	✓	✓	Permite obtener los datos de una base de datos a través de ODBC. A partir de ahí permite realizar consultas SQL y extraer tablas/vistas particulares.
Cache File	✓	✗	Para utilizar ficheros de caché creados por el propio Clementine utilizando el nodo File de la categoría Output.
SPSS Import (Import. SPSS)	✓	✓	Para importar datos de un fichero en formato SPSS (paquete estadístico).
SAS Import (Import. SAS)	✗	✓	Para importar datos de un fichero en formato SAS (paquete estadístico).
User Input	✓	✓	Nodo que no se puede insertar directamente y que sustituye a otro previamente creado, permitiendo introducir los datos manualmente.

1.2 Record Ops. (Oper. con registros)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Select (Seleccionar)	✓	✓	Permite imponer condiciones de inclusión o exclusión para las filas. Es similar a un "SELECT" en SQL.
Merge (Combinar)	✓	✓	Permite combinar los datos de dos fuentes (tablas) juntando las columnas de aquellos registros que coincidan en un atributo común de clave. Es similar a una concatenación relacional (JOIN de SQL).
Sample (Muestra)	✓	✓	Permite hacer un muestreo de los datos, ya sean de los N primeros, ir cogiendo saltados hasta N , o coger aleatoriamente un % de los registros.
Balance (Equilibrar)	✓	✓	Permite aumentar o disminuir la proporción de registros que cumplen unas determinadas condiciones, lo que es útil para <i>sobremuestrar</i> .
Sort (Ordenar)	✓	✓	Ordena los registros de un stream (como un ORDER BY de SQL).
Distinct (Distinguir)	✓	✓	Elimina los registros repetidos (como el DISTINCT de SQL pero con alguna opción más)
Aggregate (Agrega)	✓	✓	Permite aplicar funciones agregadas (como el SUM, AVG, COUNT del SQL)
Append (Añadir)	✓	✓	Permite unir (como UNION ALL de SQL) dos o más fuentes de datos.

1.3 Field Ops. (Oper. con campos)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Filter (Filtrar)	✓	✓	Permite eliminar campos no representativos o inservibles. También permite renombrar.
Derive (Derivar)	✓	✓	Permite añadir nuevos campos derivados como combinación de otros.
Type (Tipo)	✓	✓	Permite tipar los campos (si son discretos o continuos, si son de entrada o salida, etc.). Este paso suele ser necesario para poder aplicar modelos y gráficos. También se puede utilizar para descartar datos anómalos.
Filler (Rellenar)	✓	✓	Permite rellenar o sustituir campos faltantes o anómalos siguiendo unas condiciones.
History (Histórico)	✓	✓	Permite generar campos con memoria, especialmente en series (acumulados parciales, etc.)
SetToFlag (Convertir a marca)	✓	✓	Permite generar nuevos campos como transformación de un valor discreto de n posibles valores a n nuevos campos booleanos (tipo flag).

1.4 Graph (Gráficos)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Plot (Gráfico)	✓	✓	Permite representar la relación entre dos valores numéricos en dos dimensiones.
Histogram (Histograma)	✓	✓	Representa un histograma de la distribución de los datos respecto a un valor numérico.
Distribution (Distribución)	✓	✓	Representa un histograma de la distribución de los datos respecto a un valor no numérico.
Web (Malla)	✓	✓	Representan la fuerza de asociaciones entre distintos valores de dos o más atributos <i>simbólicos</i> (no numéricos). Permite ver las asociaciones entre campos.
Collect (Recolectar)	✓	✓	Similar al histograma, pero muestra la distribución de un valor numérico respecto a otro.

Directd Web (Malla direccional)	✓	✓	Genera un subconjunto de las asociaciones que muestra el nodo "Web". En este caso, muestra sólo las conexiones entre uno o más campos "FROM" a un campo "TO".
Multiplot (Gráf. múltiple)	✓	✓	Es un derivado del nodo Plot. Permite definir varios campos "Y" respecto a un campo "X"
Evaluation Chart (Evaluación)	✗	✓	Variantes de los gráficos de respuesta, que permiten evaluar qué modelo se va a comportar mejor dependiendo del contexto, como los ROI (return on investment). Muy relacionado con el análisis ROC (Receiver Operating Characteristic).

1.5 Modelling (Modelado)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Train Net (Red)	✓	✓	Red neuronal multicapa con backpropagation. Dispone de muchos parámetros. Especialmente útil para problemas de clasificación e interpolación (tanto los campos de entrada como los de salida, la clase, pueden ser simbólicos o numéricos). Requiere uno y sólo un atributo "OUT".
Build C5.0 (Crear C5.0)	✓	✓	Árbol de decisión derivado del ID3 y el C4.5 de Quinlan. Dispone de diferentes opciones. Los campos de entrada pueden ser simbólicos o numéricos, pero la clase ha de ser discreta. Requiere uno y sólo un atributo "OUT".
Tr. Kohonen	✓	✓	Redes Asociativas de Kohonen, conocidas también como Knets. Permite realizar segmentaciones (clustering). El algoritmo actúa sólo sobre los atributos definidos como IN (el resto se ignoran). El resultado es un conjunto de condiciones que separan/segmentan las instancias en dos o más grupos (clusters).
Regression (Reg. Lineal)	✓	✓	Construye un modelo de regresión lineal, es decir, una función lineal de un valor numérico respecto uno o más atributos numéricos. Requiere uno y sólo un atributo "OUT".
GRI	✓	✓	Genera reglas de asociación orientadas de uno o más atributos (numéricos o simbólicos) o un atributo simbólico de ordenadas por support y accuracy.
Train Kmeans	✓	✓	Método basado en el movimiento de centros. Permite realizar segmentaciones (clustering). El algoritmo actúa sólo sobre los atributos definidos como IN (el resto se ignoran). El resultado es un conjunto de condiciones que separan/segmentan las instancias en dos o más grupos (clusters).
A Priori	✓	✓	Este nodo descubre reglas de asociación en los datos, en la forma "if antecedent(s) then consequent(s)". Se puede especificar confianza y soporte.
Log. Regression (Reg. Logística)	✗	✓	También llamada regresión nominal. Es como la regresión lineal, pero para clasificación. Es decir, requiere uno y sólo un atributo "OUT" y debe ser discreto (nominal).
Árbol C&R	✗	✓	Es un método de clasificación y regresión basado en árboles. La clase puede ser discreta (clasificación) o continua (regresión). Requiere uno y sólo un atributo "OUT".
Factorial/PCA	✗	✓	Permite realizar "Principal component analysis (PCA)" o análisis factorial.
TwoStep	✗	✓	Un método para segmentación (clustering). Al igual que el Kmeans, el algoritmo actúa sólo sobre los atributos definidos como IN (el resto se ignoran). El resultado es un conjunto de condiciones que separan/segmentan las instancias en dos o más grupos (clusters).
Build Rule	✓	✗	Genera un conjunto de reglas de clasificación. Los campos de entrada pueden ser simbólicos o numéricos, pero la clase ha de ser discreta. Requiere uno y sólo un atributo "OUT".

1.6 Output (Salida)

La siguiente tabla recoge una breve descripción de cada uno de ellos:

NOMBRE	V5.2	V6.0	DESCRIPCIÓN
Table (Tabla)	✓	✓	Muestra los datos de un stream en una tabla.
Analysis (Análisis)	✓	✓	Se añade a la salida de un modelo para analizar su validez.
Set Globals (Def. Globales)	✓	✓	Permite calcular ciertos valores (medias, máximos, mínimos, desviaciones) que al ejecutar el nodo están disponibles para expresiones, condiciones en cualquier nodo.
Matrix (Matriz)	✓	✓	Genera una matriz de ocurrencias para los valores de dos campos. En cada una de las celdas se muestra la cantidad o el porcentaje de instancias con el par de valores de cada dimensión.
Statistics (Estadísticos)	✓	✓	Genera estadísticas de distribuciones de los distintos atributos. Especialmente útil para calcular correlaciones.
Report (Informe)	✓	✓	Permite realizar informes combinando los resultados de un stream.
File (Archivo)	✓	✓	Permite exportar a fichero, también permite exportar a ficheros cachés.
SPSS Export (Export. SPSS)	✓	✓	Exporta a ficheros de datos con el formato SPSS.
Quality (Calidad)	✓	✓	Proporciona un informe sobre la proporción de datos faltantes por campo.
ODBC Output (Salida ODBC)	✓	✓	Permite exportar y enlazar un stream con una fuente ODBC, insertando el resultado del stream en una tabla.
SPSS Procedure (Proceso SPSS)	✓	✓	Permite ejecutar in situ en el Clementine un procedimiento estadístico generado en el paquete estadístico SPSS.
Excel	✓	✓	Permite lanzar un stream al Excel.
Publisher	✓	✓	Permite exportar un modelo/condición para ser utilizado en C o a SQL.
Export. SAS	✗	✓	Exporta a ficheros de datos con el formato SAS.

Para más información sobre un nodo, pulsa en el menú "Help" → "Point-for-help Mode" y pincha en el nodo del que requieras más información.

Además de los nodos anteriores, existen nodos para los modelos, que se representan con la figura de un diamante. A partir de ahora denominaremos a estos nodos, nodos diamante, para diferenciarlos del resto.

Finalmente, la siguiente tabla resume los modelos que pueden usarse para diferentes tareas:

NOMBRE	V5.2	V6.0	PREDICTIVO		DESCRIPTIVO		
			Clasificación	Regresión	Clustering	Reglas asociación	Otros
Train Net (Red)	✓	✓	✓	✓			
Build C5.0 (Crear C5.0)	✓	✓	✓				
Tr. Kohonen	✓	✓			✓		
Regression (Reg. Lineal)	✓	✓		✓			
GRI	✓	✓				✓	
Train Kmeans	✓	✓			✓		
A Priori	✓	✓				✓	
Log. Regression (Reg. Logística)	✗	✓	✓				
Árbol C&R	✗	✓	✓	✓			
Factorial/PCA	✗	✓					✓
TwoStep	✗	✓			✓		
Build Rule	✓	✗	✓				
Statistics ² (Estadísticos)	✓	✓					✓

2. Un primer ejemplo

Vamos a construir el stream de la Figura 1. En primer lugar, aparte del Clementine, dispones de un directorio llamado "LabKDD" con todos los ficheros de datos necesarios para realizar este boletín. En particular, en este primer ejemplo, vamos a trabajar con los datos acerca de los días que se ha podido jugar al tenis, dependiendo de diversos aspectos meteorológicos. El objetivo es poder determinar (predecir) si hoy podremos jugar al tenis. Los datos de que disponemos están en el fichero: "..\LabKDD\PlayTennis\playtennis-evidence.txt" y son los siguientes:

Sky	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

² Aunque no es un modelo propiamente dicho, permite calcular correlaciones.

Abrimos el Clementine (para ello, ve a Inicio → Programas → Clementine 5.2.1 → Clementine). Al abrir el programa, las dos áreas de trabajo (izquierda superior y derecha superior) te aparecen en blanco (mejor en dicho en azul y gris).

Lo primero que vamos a hacer es insertar un nodo fuente de datos al área de trabajo. Para ello, pincha dos veces (o pincha una vez en el nodo y después otra vez en el área de trabajo) en el nodo "Var. File" que está en la categoría "Sources" (abajo a la izquierda). Te aparecerá el nodo en el área de trabajo, como se muestra en la Figura 3:

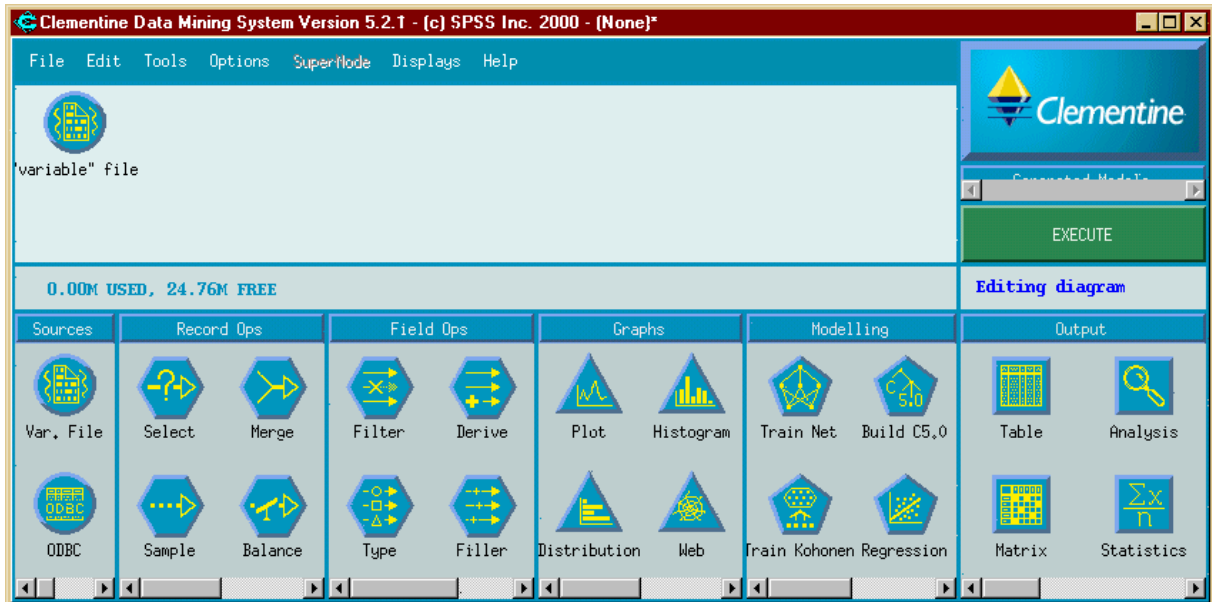


Figura 3. Insertando el primer nodo en el área de trabajo

Para borrar un nodo, simplemente se selecciona y se pulsa la tecla "Supr". También se puede borrar con el menú de contexto asociado a un nodo, el cual se abre pulsando el botón derecho sobre un nodo.

Ahora vamos a enganchar el nodo con una fuente de datos. Para ello, pincharemos con el botón derecho sobre el nodo "variable" file de la zona de trabajo y seleccionaremos "EDIT". En la pantalla de edición modificaremos el nombre del fichero, el directorio donde está y la forma de importarlo (utilizando los tabuladores). Para seleccionar el fichero, simplemente pincha en "Set File". Busca el fichero "..\LabKDD\PlayTennis\playtennis-evidence.txt" en la siguiente pantalla (Figura 4).

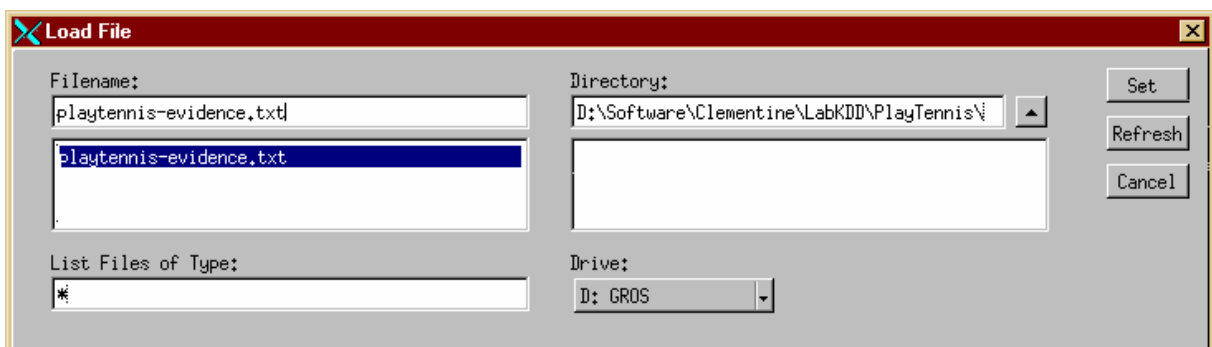


Figura 4. Buscando el fichero de datos

Pulsa "Set". A continuación pincha el CheckBox "Tab", como aparece en la siguiente Figura 5:

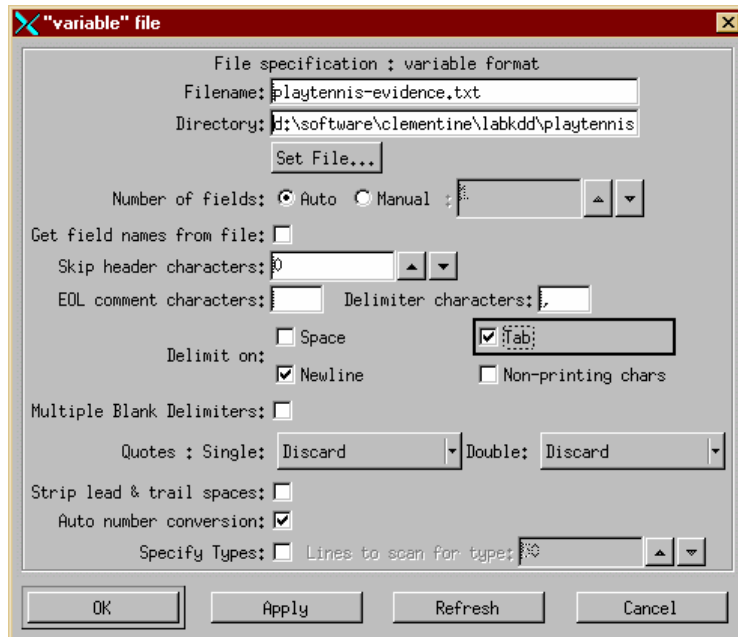


Figura 5. Enganchando con la fuente de datos

Además seleccionaremos “Get field names from file” para que nos coja el nombre de los atributos del propio fichero:

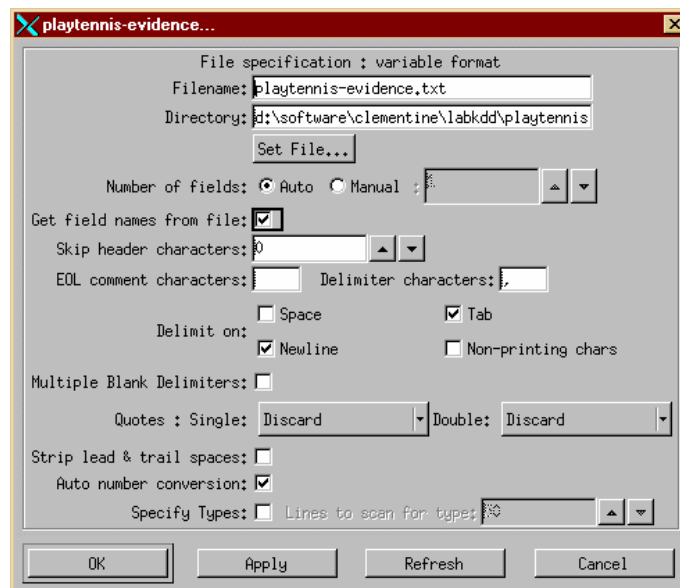


Figura 6. Opciones de formato para leer correctamente el fichero de datos

Pincha en OK.

Para ver que carga bien los datos vamos a añadir un nodo “Table” (está en la última categoría “Output”). Una vez te aparezca en la zona de trabajo hay que enganchar los dos nodos. Aquí aparece el único gran misterio del Clementine: cómo enlazar nodos.

Para enlazar dos nodos en Clementine, se han de pulsar el botón izquierdo y derecho a la vez sobre el nodo origen y arrastrar el ratón hasta el nodo destino, soltando en este momento los dos botones.
(si el ratón tiene botón del medio, también se puede utilizar este botón)

Veamos paso a paso esta delicada operación:

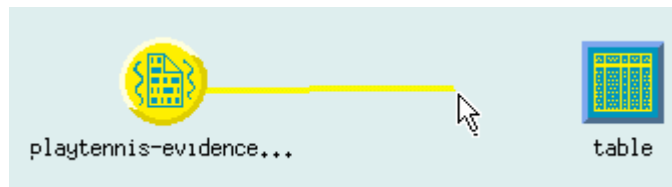


Figura 7. Enlazando dos nodos

El resultado es el siguiente:



Figura 8. Dos nodos enlazados

Para destruir un enlace, simplemente se pincha con el botón derecho en el enlace y aparece “Destroy Connection”.

Si una vez conectados, pulsas el gran botón verde de “EXECUTE” te deberán aparecer los datos importados en una tabla:

Sky	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Figura 9. Los datos se han enganchado correctamente

Como se muestra en la tabla, el problema que estábamos intentando tratar era el de ver si hoy podemos jugar al tenis. Para poder abordar este problema, hemos de decir que los campos “Sky”, “Temperature”, “Humidity”, “Wind” son predictores (es decir de entrada), mientras que el campo “Playtennis” es la clase a predecir, el resultado (es decir la salida).

Para ello vamos a añadir un nodo “Type”, que se encuentra en la categoría “Field Ops”. Ahora enlazamos el nodo “playtennis-evidence...” con el nodo “type”. Vamos a editar el nodo “type”. Para ello pinchamos con el botón derecho en “type” y pinchamos en “EDIT”. Como vemos todos los nodos tienen dirección IN. Como la salida va a ser “playtennis” modificamos su dirección a OUT.

Field	Type	Dir	Check	Blanks
Sky	Auto	IN	NONE	
Temperature	Auto	IN	NONE	
Humidity	Auto	IN	NONE	
Wind	Auto	IN	NONE	
PlayTennis	Auto	OUT	NONE	

Figura 10. Tipando los atributos.

Una vez hecho esto ya estamos en disposición de intentar aprender un modelo a partir de los datos, en este caso, una función, que dados unos determinados valores de los atributos de entrada nos dé un valor para el valor de salida.

Para ello añadimos un nuevo nodo "build C5.0" (está en la categoría de "Modelling") para construir un árbol de decisión sobre los datos. Conectamos el nodo "Type" con el nodo "build C5.0", que pasa a llamarse PlayTennis.

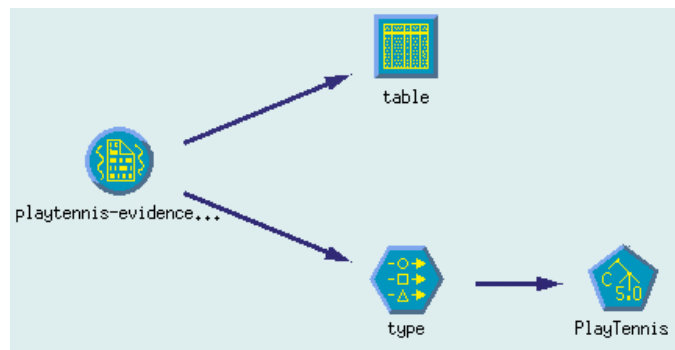


Figura 11. Stream resultante.

Ahora ya estamos en disposición de aprender un modelo (en este caso un árbol de decisión). Para ello, cogemos carrerilla y le damos bien fuerte al gran botón verde de "EXECUTE" y a minerizar!!!!

Como puedes observar, aparte de volver a mostrar la tabla con los datos de origen (pantalla que puedes cerrar para que no moleste), se ha generado un nuevo icono en el área de trabajo de la derecha, con la forma de un diamante. ¿Somos ricos?

No, de momento. Pinchamos dos veces en el diamante de la derecha y nos aparece en el área de trabajo de la izquierda.

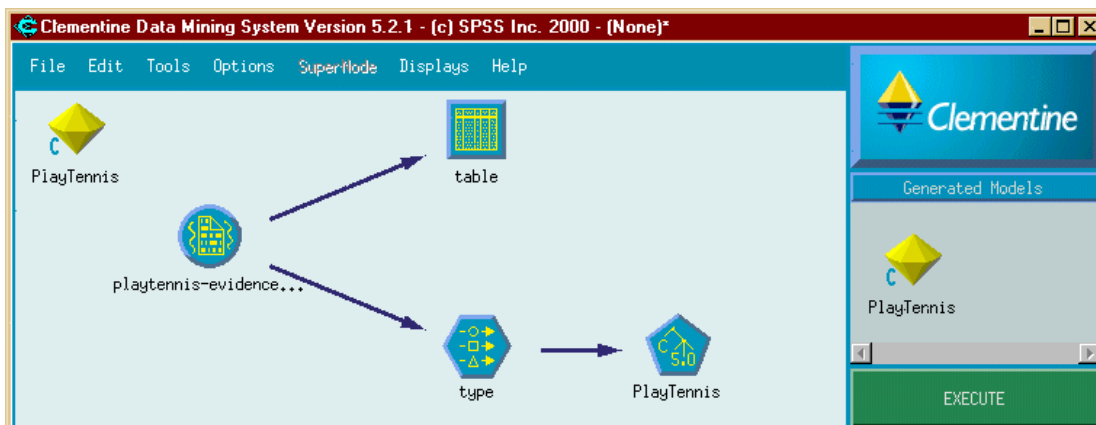


Figura 12. Modelo generado y una copia en el área de trabajo.

Ahora pinchamos con el botón derecho en el diamante que hemos copiado en el área de trabajo (el que aparece a la izquierda del todo en la Figura 12) y pinchamos en "Browse". Se nos muestra una ventana donde podemos ver el árbol de decisión creado. En el menú "View", pulsa "Show Instances/confidence". Ahora tienes el árbol etiquetado como se muestra en la siguiente figura:

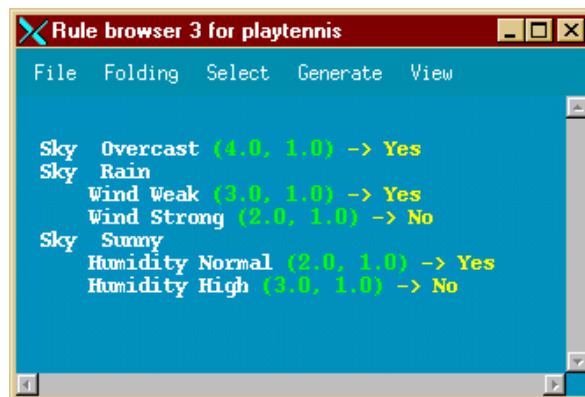


Figura 13. Árbol generado y su cobertura/confianza.

¿Cómo interpretamos el árbol anterior? La manera de verlo es la siguiente:

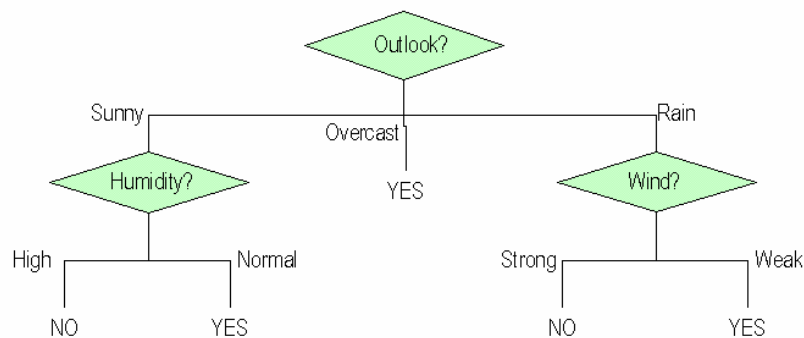


Figura 14. Árbol representado gráficamente.

Además, la Figura 13 nos muestra para cada rama cuántos ejemplos de la evidencia son cubiertos y con qué confianza (en este caso el 100% en todas las ramas).

La representación lógica del árbol anterior sería:

$(\text{Outlook}=\text{Sunny AND Humidity}=\text{Normal}) \text{ OR } (\text{Outlook}=\text{Overcast}) \text{ OR } (\text{Outlook}=\text{Rain AND Wind}=\text{Weak})$

Podemos verlo también en forma de reglas. Para ello, si le damos al botón derecho del ratón sobre el nodo-diamante "Playtennis" y seleccionamos "Browse", podemos, en el menú "Generate", generar las reglas para ese árbol de decisión, como se muestra en la siguiente figura:

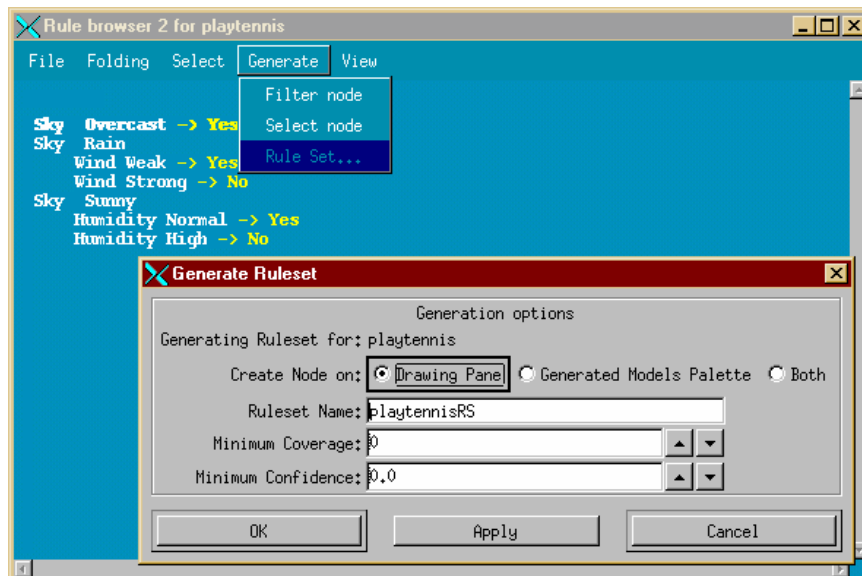


Figura 15. Generando las reglas del árbol.

Esto nos genera un nuevo nodo “playtennisRS” que si lo observamos (botón derecho y “Browse”) veremos la expresión del árbol en forma de reglas. Dándole al menu “Folding” y después a “Show More” vemos:

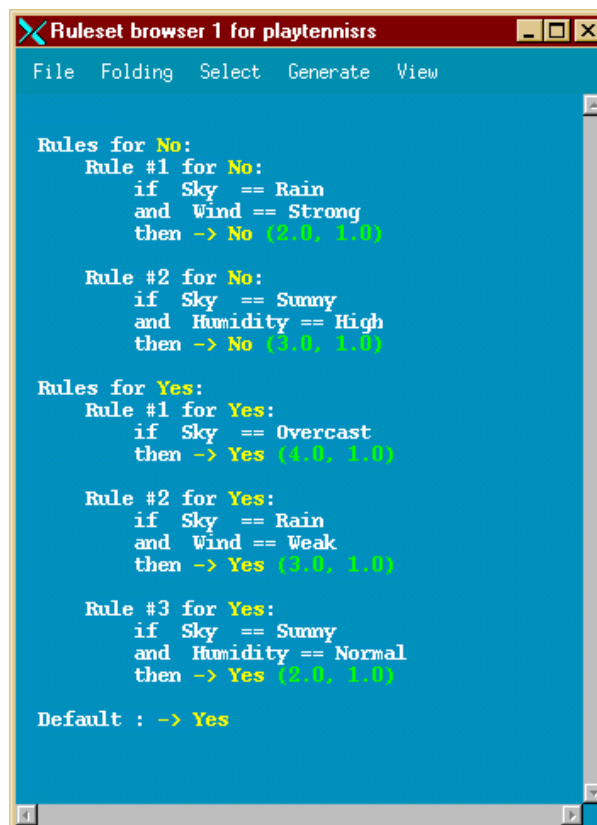


Figura 16. Reglas del árbol

Finalmente, podemos utilizar el modelo para predecir si podemos jugar o no jugar al tenis hoy. Supongamos que tenemos, p.ej., la instancia (Sky = rain, Temperature = cool, Humidity = high, Wind = strong). Podemos ver en el árbol que la clase es NO, con lo que no jugaremos al tenis. No nos haremos ricos pero nos ahorraremos un constipado o una rinitis alérgica.

Podemos grabar lo que hemos realizado en un fichero .str mediante “File → Save Stream”.

3. Visualización y Preparación de Datos

Vamos a abordar problemas más complejos e interesantes a partir de ahora.

3.1 Enunciado de un Primer Problema. Selección de Fármaco

En este caso se trata de predecir el tipo de fármaco (drug) que se debe administrar a un paciente afectado de rinitis alérgica según distintos parámetros/variables. Las variables que se recogen en los historiales clínicos de cada paciente son:

- Age: Edad
- Sex: Sexo
- BP (Blood Pressure): Tensión sanguínea.
- Cholesterol: nivel de colesterol.
- Na: Nivel de sodio en la sangre.
- K: Nivel de potasio en la sangre.

Hay cinco fármacos posibles: DrugA, DrugB, DrugC, DrugX, DrugY. Se han recogido los datos del medicamento idóneo para muchos pacientes en cuatro hospitales (los ficheros están en el directorio “..\LabKDD\drugs”). Se pretende, para nuevos pacientes, determinar el mejor medicamento a probar a cada uno.

3.2 Resolución del Problema de Selección de Fármaco

En primer lugar vamos a coger los datos del primer hospital, ya que al ser el de menor tamaño (200 registros), permite hacer más pruebas inicialmente. Limpiamos la zona de trabajo (File → Clear) o (File → Close). Los datos del fichero “drug1n” tienen cabecera, con lo que añadiremos un nodo fuente “Var. File” y al editar, indicaremos el nombre y directorio del fichero y marcaremos el checkbox para que obtenga el nombre de los atributos a partir de la primera línea del fichero:

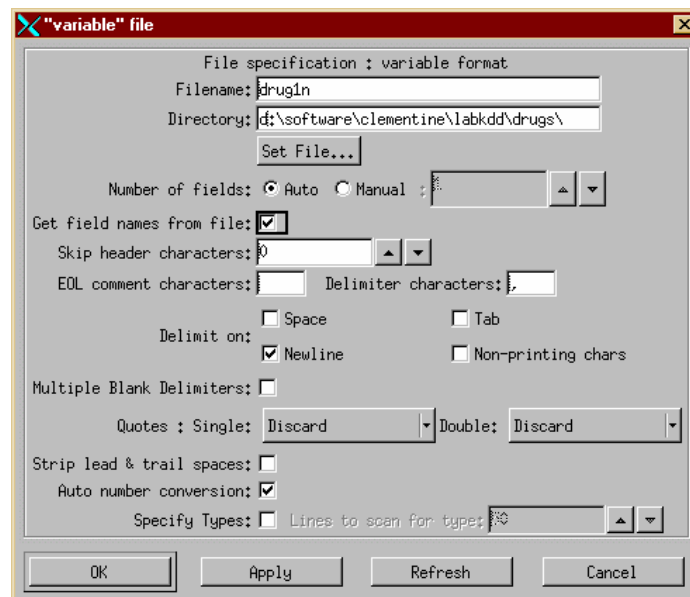


Figura 17. Abriendo el fichero del primer hospital

Ahora añadiremos un nodo tabla y lo engancharemos con el nodo “Var. File”. Podemos ver los datos ya cargados al ejecutar la tabla, como se ve en la siguiente figura:

Age	Sex	BP	Cholesterol	Na	K	Drug
23	F	HIGH	HIGH	0.793	0.031	drugY
47	M	LOW	HIGH	0.739	0.056	drugC
47	M	LOW	HIGH	0.697	0.069	drugC
28	F	NORMAL	HIGH	0.564	0.072	drugX
61	F	LOW	HIGH	0.559	0.031	drugY
22	F	NORMAL	HIGH	0.677	0.079	drugX
49	F	NORMAL	HIGH	0.79	0.049	drugY
41	M	LOW	HIGH	0.767	0.069	drugC
60	M	NORMAL	HIGH	0.777	0.051	drugY
43	M	LOW	NORMAL	0.526	0.027	drugY

Figura 18. Datos del primer hospital

La primera pregunta que nos podemos hacer es ver qué fármacos son más comunes en general, para ver si todos suelen ser igualmente efectivos en términos generales. Para ello añadimos un nodo "Distribution" en la categoría "Graph" y lo enlazamos con la fuente de datos "drug1n". La situación debe ser similar a la siguiente:

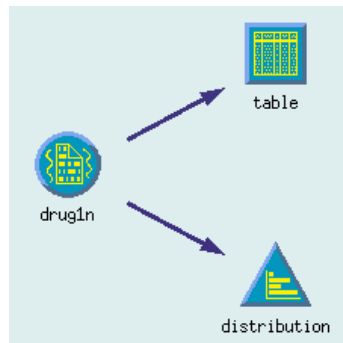


Figura 19. Primeros pasos analizando los fármacos

Si editamos el nodo "distribution", podemos elegir el atributo por el cual vamos a representar el gráfico. Seleccionamos "drug" como el atributo para mostrar las distribuciones.

Figura 20. Pasos para visualizar la distribución de los fármacos

Ahora ya podemos ejecutar el "stream" y ver la distribución del uso de fármacos en el hospital 1.

Value	Proportion	%	Occurrences
drugA		11.5	23
drugB		8.0	16
drugC		8.0	16
drugX		27.0	54
drugY		45.5	91

Figura 21. Distribución de los fármacos en el hospital 1

Vemos que el fármaco más efectivo es el Y, que se administra con éxito en casi la mitad de los pacientes. Una regla vulgar sería aplicar el fármaco Y, en el caso que falle, el fármaco X, y así sucesivamente siguiendo las frecuencias de uso con éxito.

Con la herramienta Clementine, seguro que lo podemos hacer mucho mejor...

Apliquemos lo mismo que hemos realizado en el ejemplo anterior, intentemos generar un árbol de decisión. Construyamos y ejecutemos un stream con un nodo "Type" (donde el atributo drug se ha puesto como salida) y con un nodo C5.0. Añadamos el modelo generado al área de trabajo como se ve en la siguiente figura:

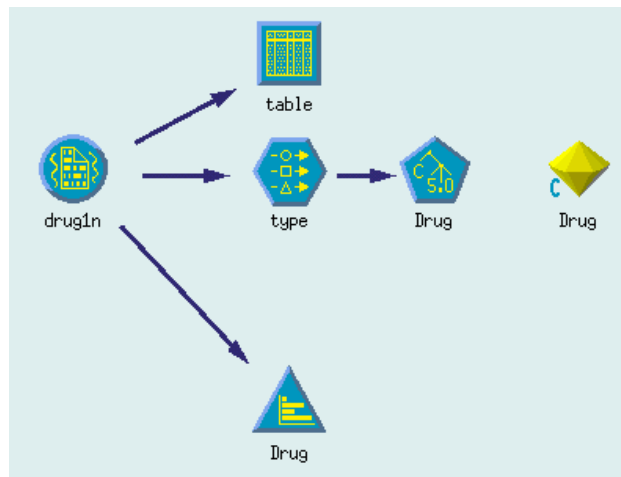


Figura 22. Realización de un modelo directamente sobre los datos

El modelo resultante es el siguiente:

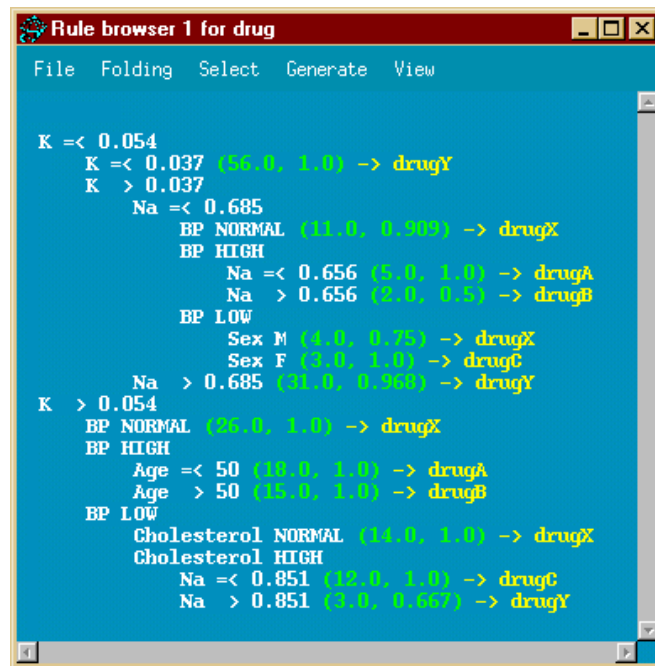


Figura 23. Árbol de decisión directamente sobre los datos

Como podemos observar, el árbol tiene bastante ramas (en concreto 13). Podemos ver cuál es la precisión (accuracy) de este árbol respecto a los datos de entrenamiento. Para ello, conectamos el nodo "Type" al nodo diamante "Drug" y éste a un nuevo nodo "Analysis" de la categoría "Output", como se ve en la siguiente figura:

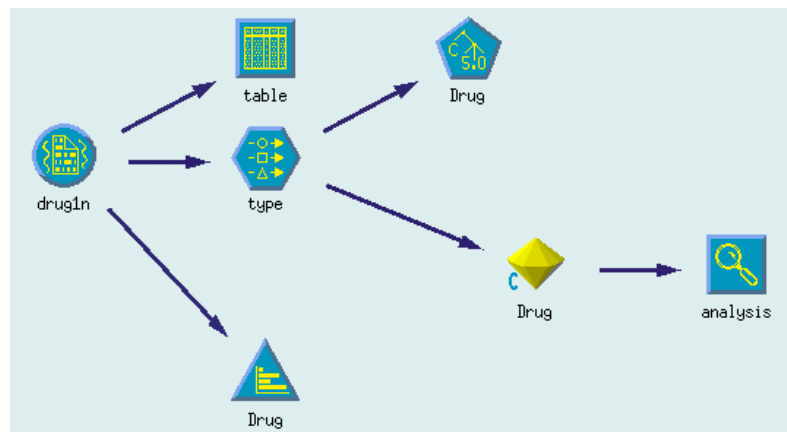


Figura 24. Stream para analizar la calidad de un modelo

Si ejecutamos el nodo "analysis" (con el menú contextual pulsando el botón derecho) obtenemos los siguientes resultados:

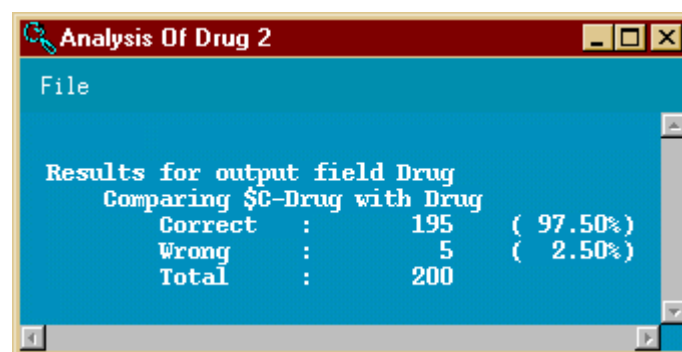


Figura 25. Calidad del modelo generado para los datos de entrenamiento

Con lo que tenemos un 97,5% de precisión. Es decir un error de sólo el 2,5% sobre los datos de entrenamiento. Este modelo es muchísimo mejor que si sólo nos guiamos por la distribución, que nos daría un error de más del 50%.

De todas maneras, es posible hacerlo mejor... ¿pero cómo? ¿con otro tipo de algoritmo de aprendizaje, una red neuronal, p.ej.?

Es posible que otros modelos (p.ej. las redes neuronales) dieran mejor resultado, pero el asunto aquí es que igual no hemos examinado suficientemente los datos de entrada.

Vamos a analizar, con más detenimiento, los atributos de entrada del problema. Es posible que se puedan establecer mejores modelos si combinamos algunos atributos. Podemos analizar pares de atributos utilizando diferentes gráficos.

Para comparar los atributos discretos, el gráfico “Web” suele ser conveniente. Creemos un nodo “web” y enganchemos el nodo de fuentes de datos “drug1n” con él. Al editarlo, decimos que vamos a examinar los campos (web fields) discretos (Sex, BP, Cho, Drug):

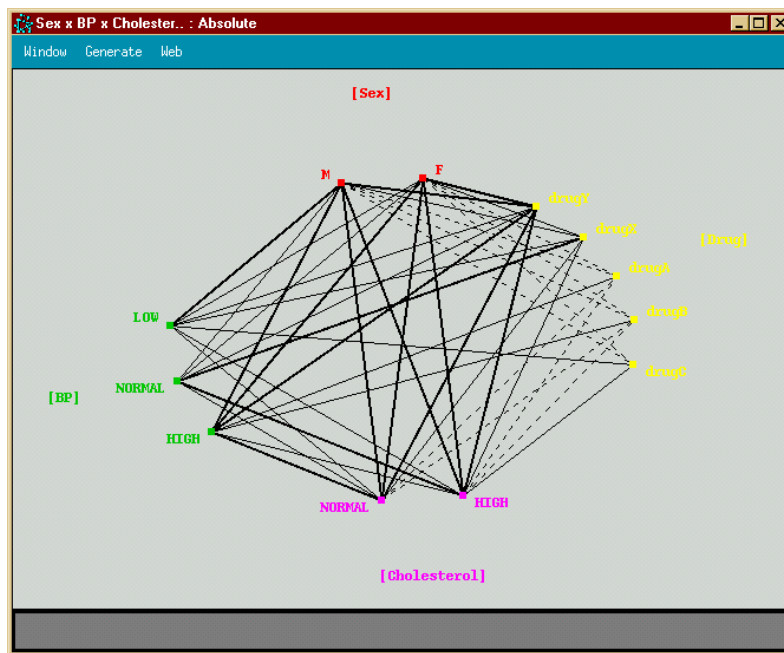


Figura 26. Relación (asociaciones) entre los atributos discretos Sex x BP x Cho x Drug

Las líneas más gruesas representan asociaciones más fuertes. No parece ver ninguna relación especial entre los distintos valores de los atributos discretos, exceptuando la clase, que va decreciendo la intensidad de una manera regular para los fármacos menos usuales.

Estudiemos la relación que hay entre los atributos continuos y su influencia en la clase. Para ello vamos a utilizar el gráfico “plot” de la categoría “graph”. Añadamos un nodo “plot” y comparemos la presión sanguínea y el colesterol. Para ello al editar el nodo “plot” pondremos en el *X field* el campo Na, en el *Y field* el campo K y en el *Overlay field* la clase Drug. Lo demás se deja por defecto, como se ve en la siguiente figura:

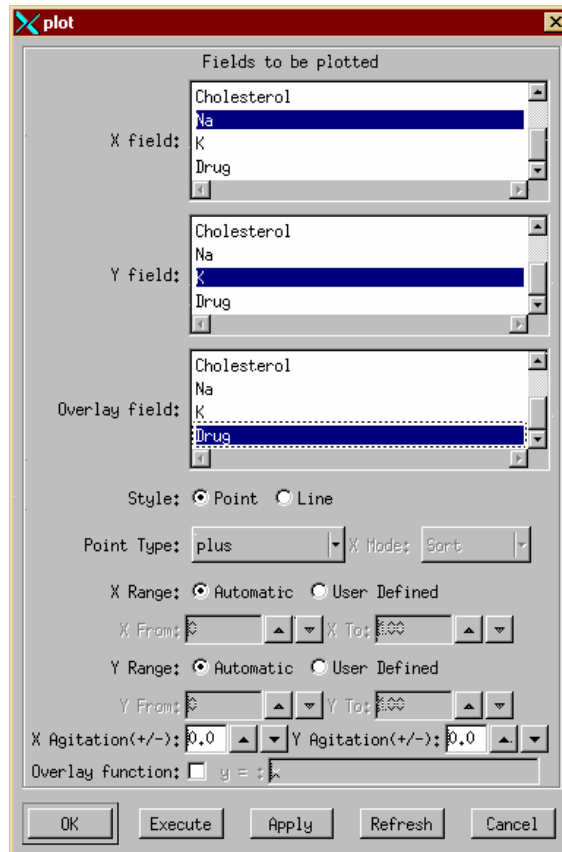


Figura 27. Editando un nodo Plot

El resultado al ejecutar el gráfico es el siguiente:

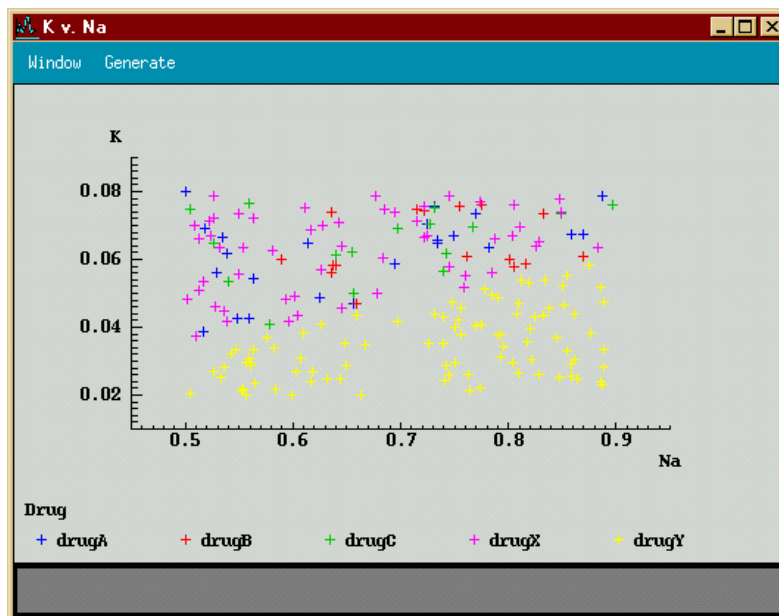


Figura 28. Resultado de un Plot (Na x K x Drug)

En este gráfico sí que se ven algunas características muy significativas. Parece haber una clara separación *lineal* entre una relación K/Na alta y una relación K/Na baja. De hecho, para las concentraciones K/Na bajas, el fármaco Y es el más efectivo de una manera clara y parece mostrarse que por encima de un cierto cociente K/Na ese medicamento deja de ser efectivo y se debe recurrir a los otros cuatro.

Podemos utilizar este conocimiento que acabamos de extraer para mejorar nuestros modelos. **Hemos establecido que el medicamento a administrar depende en gran medida del cociente entre K/Na.** Por tanto, vamos a realizar un nuevo modelo que utilice este cociente. Para ello, vamos a crear un nuevo atributo derivado (también llamados atributos pick & mix) mediante el nuevo nodo "Derive" de la categoría "Field Ops", que engancharemos con el nodo drug1n.

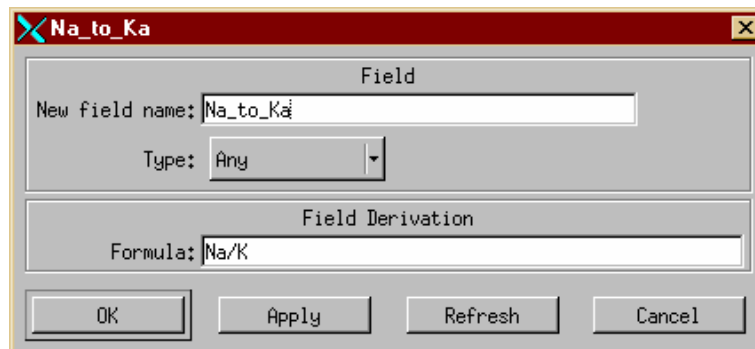


Figura 29. Definiendo un atributo derivado

NOTA: El Clementine trabaja con el teclado americano. Las teclas más necesarias son las siguientes:

- `_`: está en el ?
- `/, *, -, +`: usa el teclado numérico
- `< >`: están en , y en .
- `&`: está en el 7
- `|`: está en el >

Ahora duplica (para duplicar un nodo, en el menú contextual con el botón derecho pulsamos "Duplicate") los nodos Type y el nodo C5.0 Drug que teníamos de antes y engánchalos al stream que sale del nodo Na_to_Ka. Cambia el nombre del nodo Drug (C5.0) a Drug2, para no liarte. Para ello edita el nuevo nodo Drug y cambia el campo "Output Name":

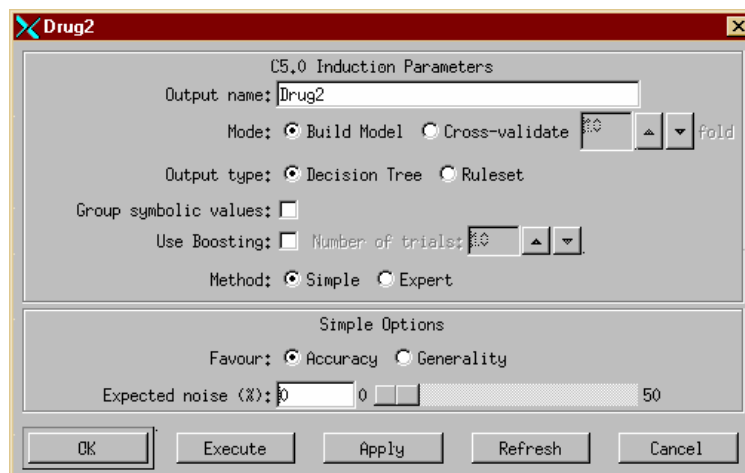


Figura 30. Cambiando el nombre del modelo a generar

Ahora ejecuta ese nodo y te generará un nuevo modelo "Drug2". Añádelo al área de trabajo, engánchalo con el nodo "Copy of Type" y añade un nodo de "Analysis" como hicimos con el primer modelo. El grafo de streams resultante debe ser el de la siguiente figura:

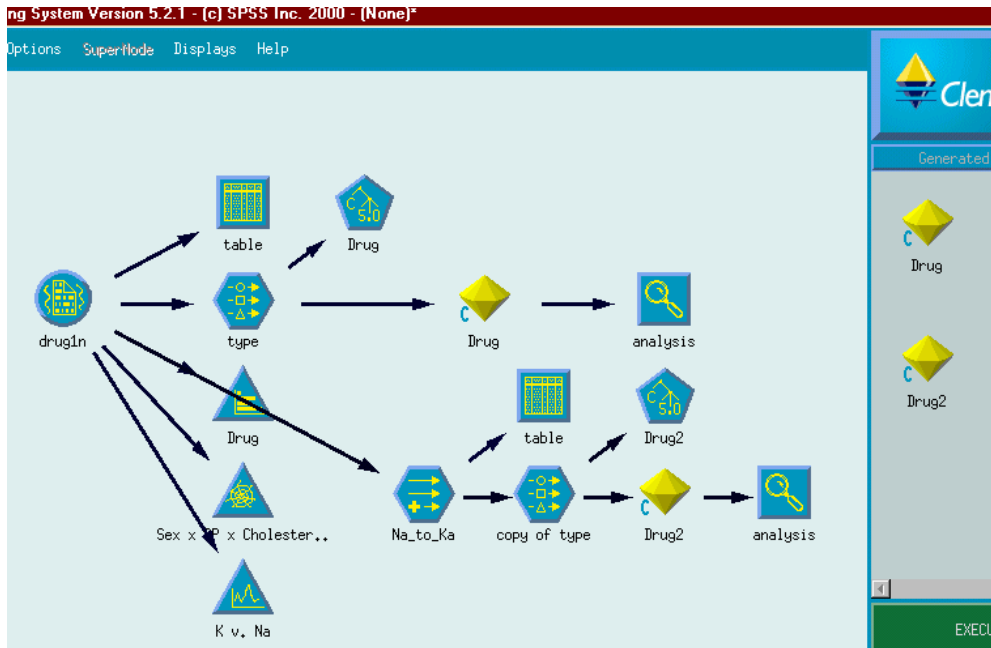


Figura 31. Streams realizados para el problema del fármaco

Ahora, si examinamos el modelo Drug2 tenemos lo siguiente:

```

Rule browser 1 for drug2
File Folding Select Generate View

Na_to_Ka <= 14.642
BP NORMAL (30.0, 1.0) -> drugX
BP HIGH
  Age <= 50 (20.0, 1.0) -> drugA
  Age > 50 (10.0, 1.0) -> drugB
BP LOW
  Cholesterol NORMAL (10.0, 1.0) -> drugX
  Cholesterol HIGH (10.0, 1.0) -> drugC
Na_to_Ka > 14.642 (20.0, 1.0) -> drugY
  
```

Figura 32. Segundo modelo que utiliza el atributo derivado Na_to_K

Modelo mucho más simple y corto que el anterior, en el que se ve la importancia del atributo que hemos creado Na_to_K.

Además si analizamos su calidad con el nodo de análisis, tenemos:

```

Analysis Of Drug 5
File

Results for output field Drug
Comparing $C-Drug with Drug
Correct : 200 (100.00%)
Wrong : 0 ( 0.00%)
Total : 200
  
```

Figura 33. Calidad del segundo modelo que utilizar el atributo derivado Na_to_K

Tenemos con el nuevo modelo un 100% de precisión, con lo que el modelo es mucho más fiable que antes.

Ahora graba el stream en un fichero “.str”, p.ej. “drug1.str”, porque volveremos sobre este problema.

3.3 Un Segundo Problema: Agrupación de Empleados

La empresa de software para Internet “MemolunWeb” quiere extraer tipologías de empleados, con el objetivo de hacer una política de personal más fundamentada y seleccionar a qué grupos incentivar. Las variables que se recogen de las fichas de los 15 empleados de la empresa son:

- Sueldo: sueldo anual en euros.
- Casado: si está casado o no.
- Coche: si viene en coche a trabajar (o al menos si lo aparca en el parking de la empresa).
- Hijos: si tiene hijos.
- Alq/Prop: si vive en una casa alquilada o propia.
- Sindic.: si pertenece al sindicato revolucionario de Internet
- Bajas/Año: media del nº de bajas por año
- Antigüedad: antigüedad en la empresa
- Sexo: H: hombre, M: mujer.

Los datos de los 15 empleados se encuentran en el directorio “..\LabKDD\empleados”). Se intenta extraer grupos de entre estos quince empleados.

3.4 Resolución del Problema de Agrupación de Empleados

En primer lugar vamos a leer los datos de los empleados. Limpiamos la zona de trabajo (File → Clear) o (File → Close). Los datos del fichero “empleados” tienen cabecera y están tabulados, con lo que añadiremos un nodo fuente “Var. File” y al editar, indicaremos el nombre y directorio del fichero y marcaremos el checkbox para que obtenga el nombre de los atributos a partir de la primera línea del fichero y además marcaremos el checkbox Tab:

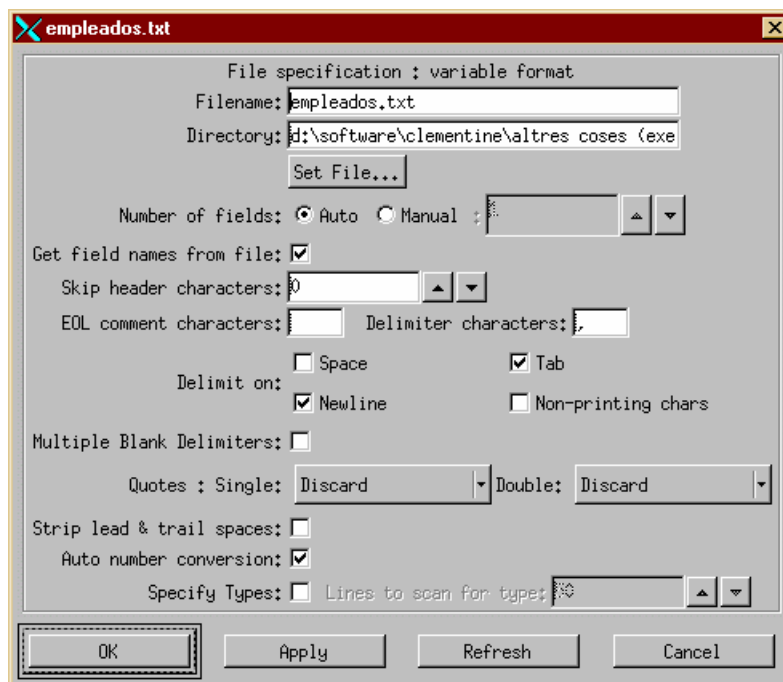


Figura 34. Abriendo el fichero fuente empleados.txt

A continuación añadimos un nodo “Type” y lo enganchamos. Todos los atributos deberían estar IN exceptuando el número de ejemplo, que es irrelevante y pondremos NONE, como se muestra en la siguiente figura:

Field	Type	Dir	Check	Blanks
#Ej	Auto	NONE	NONE	
Sueldo	Auto	IN	NONE	
Casado	Auto	IN	NONE	
Coche	Auto	IN	NONE	
Hijos	Auto	IN	NONE	
Alq/Prop	Auto	IN	NONE	
Sindic.	Auto	IN	NONE	
Bajas/Año	Auto	IN	NONE	
Antigüedad	Auto	IN	NONE	
Sexo	Auto	IN	NONE	

Figura 35. Tipando los datos de los empleados

Ahora vamos a utilizar un algoritmo de clustering para obtener grupos sobre esta población. En primer lugar vamos a probar con tres grupos. Para ello añadimos un nodo Kmeans, lo enganchamos al nodo Type y modificamos el campo "Number of clusters" a 3, como se ve en la siguiente figura:

K-Means parameters

Model name: Kmeans

Number of clusters: 3

Stop On: Default Iterations Change

Number of Iterations: [spinners]

Change Tolerance: [spinners]

Generate distance field:

Cluster display: String Number

Show cluster proximity:

Expert:

Buttons: OK, Execute, Apply, Refresh, Cancel

Figura 36. Determinando el número de clusters

Ahora podemos ejecutar el stream, obteniendo un nodo diamante Kmeans. Lo podemos añadir y enganchar al nodo Type, como se muestra en la siguiente figura:

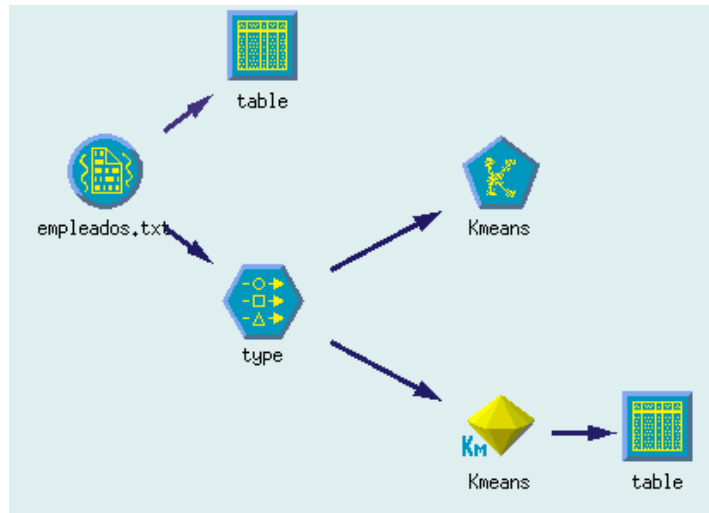


Figura 37. Stream para el problema de los empleados

Si examinamos (Browse) el nodo diamante Kmeans vemos qué características tiene cada cluster:

cluster 1	cluster 2	cluster 3
5 examples	4 examples	6 examples
Sueldo : 226000	Sueldo : 225000	Sueldo : 188333
Casado : No -> 0.8	Casado : No -> 1.0	Casado : Sí -> 1.0
Sí -> 0.2	Coche : Sí -> 1.0	Coche : Sí -> 1.0
Coche : No -> 0.8	Hijos : 0	Hijos : 2
Sí -> 0.2	Alq/Prop : Alquiler -> 0.75	Alq/Prop : Alquiler -> 0.17
Hijos : 0	Prop -> 0.25	Prop -> 0.83
Alq/Prop : Alquiler -> 1.0	Sindic. : Sí -> 1.0	Sindic. : No -> 0.67
Sindic. : No -> 0.8	Bajas/Año : 2	Sí -> 0.33
Sí -> 0.2	Antigüedad : 8	Bajas/Año : 5
Bajas/Año : 8	Sexo : H -> 0.25	Antigüedad : 8
Antigüedad : 8	M -> 0.75	Sexo : H -> 0.83
Sexo : H -> 0.6		M -> 0.17

Si añadimos una tabla a la salida del nodo diamante Kmeans podemos observar qué ejemplos exactamente han caído en qué clúster.

#Ej	Sueldo	Casado	Coche	Hijos	Alq/Prop	Sindic.	Bajas/Año	Antigüedad	Sexo	\$KM-Kmeans	\$KMD-Kmeans
1	100000	Sí	No	0	Alquiler	No	7	15	H	cluster-1	1.045
2	200000	No	Sí	1	Alquiler	Sí	3	3	M	cluster-2	0.512
3	150000	Sí	Sí	2	Prop	Sí	5	10	H	cluster-3	0.728
4	300000	Sí	Sí	1	Alquiler	No	15	7	M	cluster-3	1.324
5	100000	Sí	Sí	0	Prop	Sí	1	6	H	cluster-3	0.944
6	400000	No	Sí	0	Alquiler	Sí	3	16	M	cluster-2	0.696
7	250000	No	No	0	Alquiler	Sí	0	8	H	cluster-1	0.983
8	200000	No	Sí	0	Prop	Sí	2	6	M	cluster-2	0.804
9	200000	Sí	Sí	3	Prop	No	7	5	H	cluster-3	0.63
10	300000	Sí	Sí	2	Prop	No	1	20	H	cluster-3	0.814
11	500000	No	No	0	Alquiler	No	2	12	M	cluster-1	0.994
12	80000	Sí	Sí	2	Prop	No	3	1	H	cluster-3	0.629
13	200000	No	No	0	Alquiler	No	27	5	M	cluster-1	1.011
14	100000	No	Sí	0	Alquiler	Sí	0	7	H	cluster-2	0.854
15	80000	No	Sí	0	Alquiler	No	3	2	H	cluster-1	1.07

Figura 38. Ejemplos agrupados por clusters.

¿Cómo interpretarías los tres grupos anteriores?

Varía el número de clusters (2, 4, 5, ...) y vuelve a ejecutar el stream para cada uno de estos valores. ¿qué se puede observar?

Ahora graba el stream en un fichero “.str”, p.ej. “empleados.str”, porque volveremos sobre este problema.

3.5 Obtención y Transformación de Datos Relacionales

Aunque en la mayoría de ejemplos vamos a trabajar por comodidad a partir de datos que están en ficheros, en la mayoría de aplicaciones reales los datos originales se encuentran en una base de datos. Clementine permite enlazar a través de ODBC con cualquier motor de un Sistema de Gestión de Bases de Datos (SGBD).

Pasemos a trabajar con una base de datos clásica, el ejemplo “Neptuno” en MsAccess que versa sobre una compañía de pedidos. La base de datos se encuentra en el directorio “..\LabKDD\Neptuno”. Si abres la base de datos “Neptuno.mdb”, la estructura de la base de datos es la siguiente:

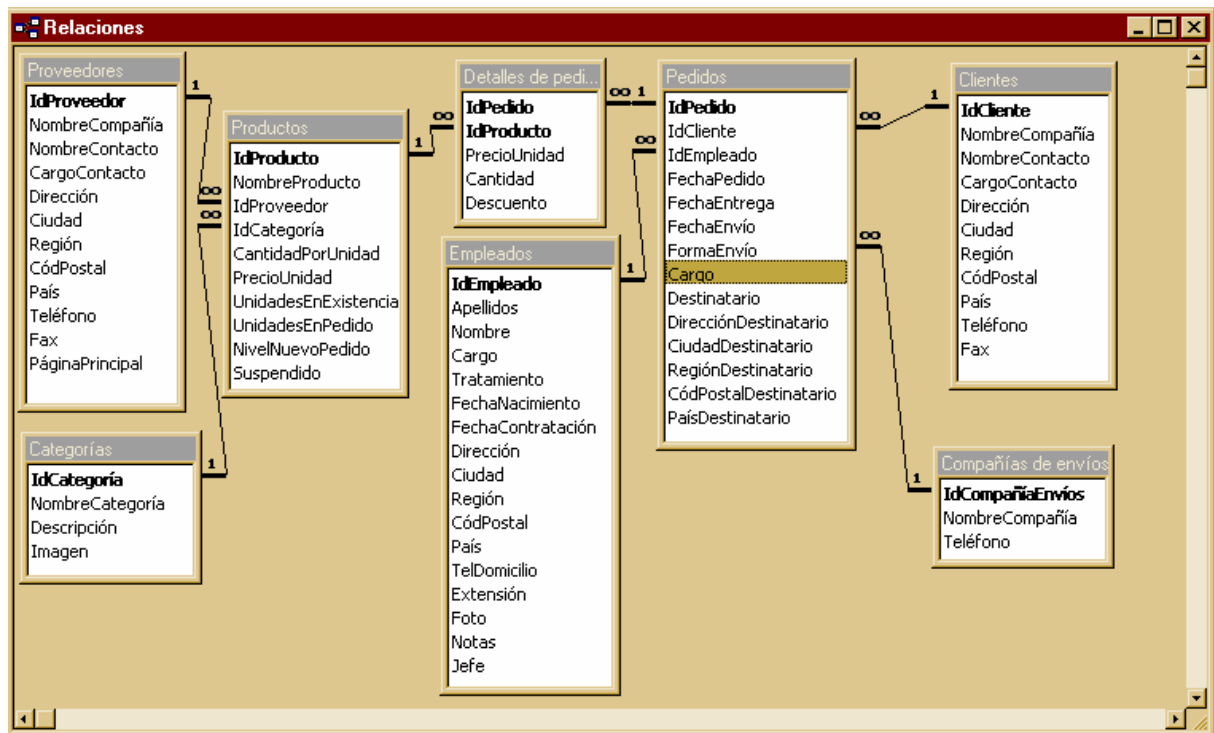


Figura 39. Base de Datos Neptuno

Sin entrar en más detalles en el esquema, el problema que vamos a tratar en este caso es el de predecir el volumen de pedidos para el próximo cuatrimestre.

Para ello, se ha realizado una consulta específica para esta cuestión que obtiene el total de los cargos de los pedidos por cuatrimestre. Dicha consulta (en realidad es una vista) se llama “_VentasPorCuatrimestre”:

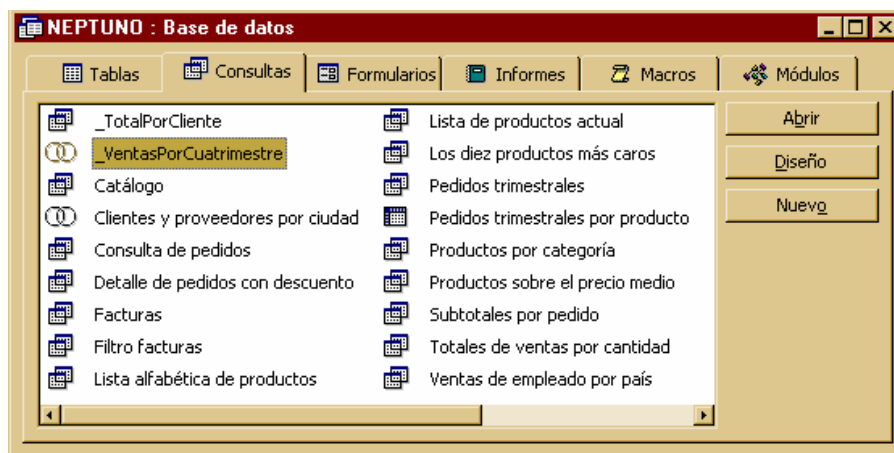


Figura 40. Vistas de la Base de Datos

Si la ejecutamos, tenemos el siguiente resultado:

ORD	ANYO	TOTAL
1	1994A	
2	1994B	
3	1994C	2.413 pta
4	1994D	4.404 pta
5	1995A	6.356 pta
6	1995B	7.547 pta
7	1995C	8.458 pta
8	1995D	9.282 pta
9	1996A	12.999 pta
10	1996B	12.496 pta
11	1996C	
12	1996D	

Figura 41. Ventas por Cuatrimestre

Nos interesa realizar una estimación para los cuatrimestres 1996C y 1996D. Para ello, vamos a utilizar el Clementine.

Antes debemos crear una fuente ODBC en el sistema. Para ello (en Windows) vamos al Panel de Control y elegimos “Fuentes de Datos ODBC”³ (en los laboratorios es posible que no esté el panel de control disponible. Entonces busca y lánzalo desde “Inicio → Oracle → Oracle para Windows 95 → Ms ODBC Administrator”):

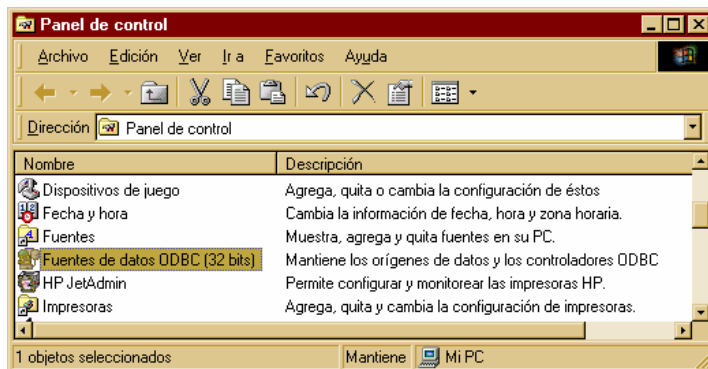


Figura 42. Panel de Control

Pinchamos y, en la pestaña “DSN de Usuario” agregamos un origen “Base de Datos de Ms Access”:

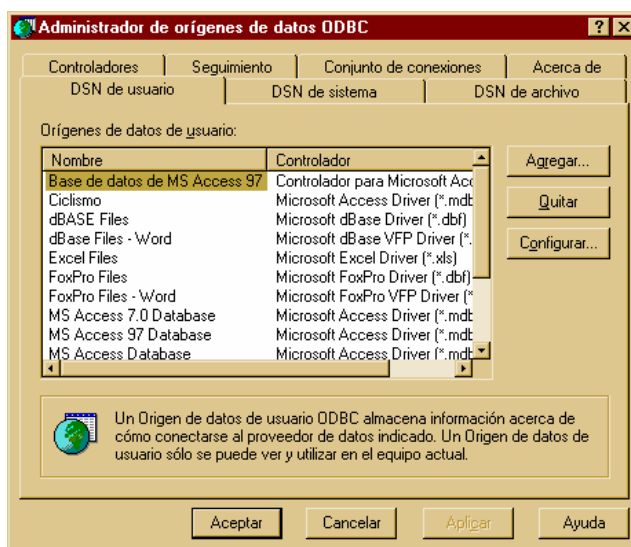


Figura 43. Panel de Control

³ En Windows 2000 está en “Panel de Control → Herramientas Administrativas → Orígenes de datos (ODBC)”.

En la siguiente pantalla “Crear nuevo origen de datos”, elige “Controlador para Microsoft Access (*.mdb)” y pincha “Finalizar”. En la siguiente pantalla pincha en “Seleccionar” y selecciona el fichero “..\LabKDD\Neptuno\Neptuno.mdb”. Le daremos el nombre del origen de datos “Fuente_Neptuno” y la descripción que quieras.

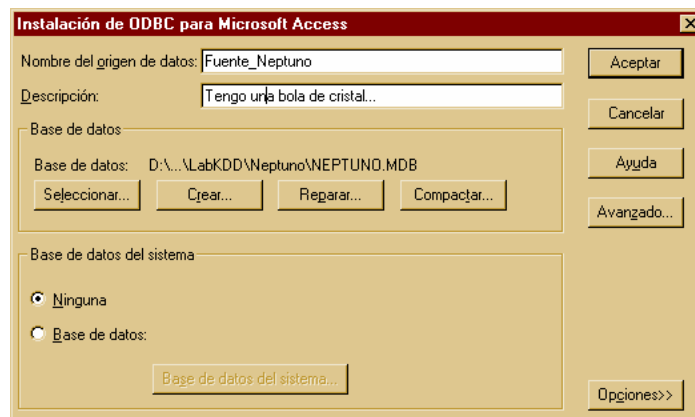


Figura 44. Panel de Control

Ahora volvemos al Clementine. Limpiamos el área de trabajo y añadimos un nodo ODBC de la categoría SOURCE. Si lo editamos, y apretamos el botón “Connect” aparece otra pantalla donde podemos seleccionar la fuente “Fuente_Neptuno” y pulsa “Connect” (no hace falta ni username ni password):

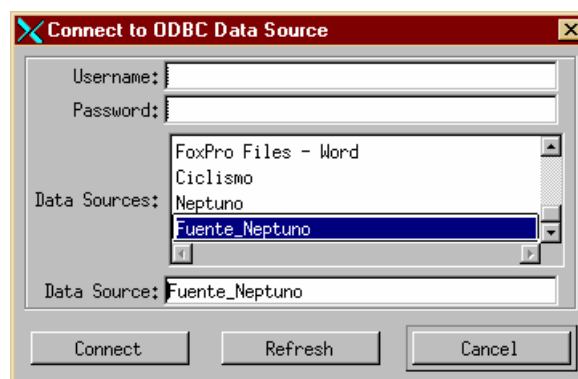


Figura 45. Conectando el nodo ODBC con la Fuente ODBC

Ahora te permite seleccionar una tabla o realizar una consulta. Pulsa “Select Table/View” y selecciona “_VentasPorCuatrimestre”. Te debe aparecer lo que se muestra en la siguiente figura:

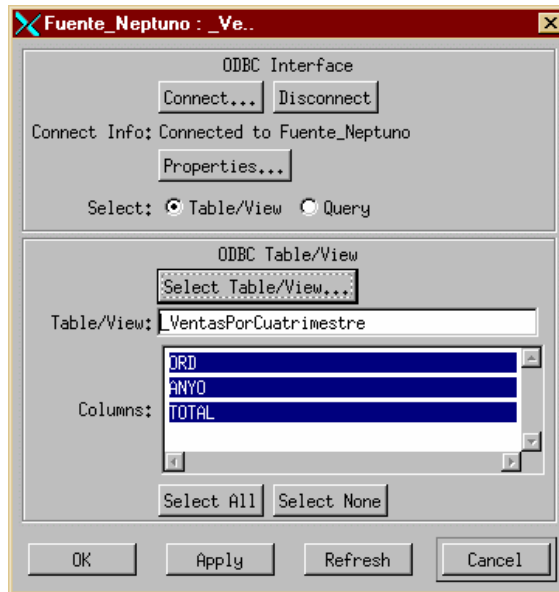


Figura 46. Conectando el nodo ODBC con la Fuente ODBC

Dale a "OK". Ahora añade un nodo "Table" al área de trabajo y conéctalo con la fuente de datos. Ejecuta el STREAM. El resultado será el que se ve en la siguiente figura:

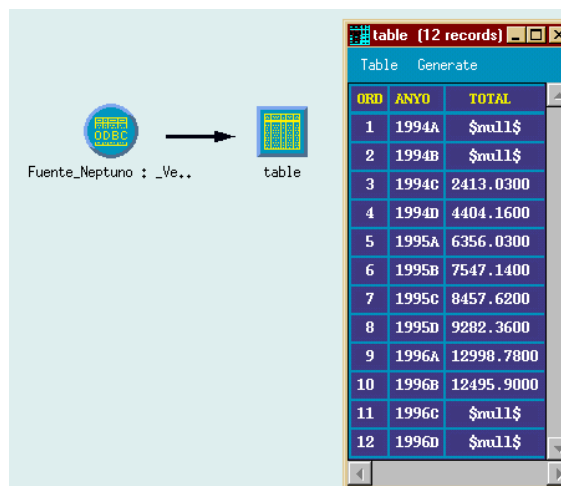


Figura 47. Datos conectados a través de ODBC

Ahora, en primer lugar debemos descartar los valores nulos, ya que no nos van a servir para predecir. Esto se puede hacer con un nodo "Type". Si lo enganchamos al nodo "Fuente_Neptuno", podemos indicar lo siguiente:

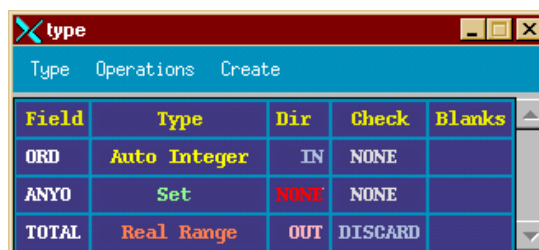


Figura 48. Tipando e indicando entrada y salida

Con la opción "DISCARD" descartamos aquellos que no sean del tipo especificado, en este caso reales. Además hemos eliminado ANYO (dirección = NONE) porque utilizaremos ORD como valor numérico para los años.

En segundo lugar, parece observarse que los datos del trimestre 1996B pueden no estar acabados y los datos del trimestre 1994C pueden ser incompletos. Lo mejor es ignorarlos. Para ello, necesitamos añadir nodos “select” para quitarlos. Existe una manera muy cómoda de hacerlo. Sobre los propios datos de la tabla que genera el nodo tabla cuando se ejecuta señalamos aquellas filas que no nos interesan, como se ve en la siguiente figura:

ORD	ANYO	TOTAL
1	1994A	\$null\$
2	1994B	\$null\$
3	1994C	2413.0300
4	1994D	4404.1600
5	1995A	6356.0300
6	1995B	7547.1400
7	1995C	8457.6200
8	1995D	9282.3600
9	1996A	12998.7800
10	1996B	12495.9000
11	1996C	\$null\$
12	1996D	\$null\$

Figura 49. Generando una condición automáticamente

Seleccionamos el menú “Generate” → “Select Node (“Records”)”. Automáticamente nos genera un nodo “Select”. Lo conectamos al nodo “Type”. Si editamos el nodo “Generate” vemos que nos aparecen justamente las condiciones para incluir esas dos filas. Modificamos el nodo “Select” de tal manera que nos excluya los cuatrimestres que no queremos, quedando de la siguiente manera:

Record Selection Criterion

Mode: Include Discard

Condition: ('ANYO' = '1994C') or ('ANYO' = '1996B')

Buttons: OK, Apply, Refresh, Cancel

Figura 50. Modificando una condición

Si conectamos un nuevo nodo “Table” a la salida del nodo “Select” y éste al nodo “Type” podremos ver que ya sólo tenemos los datos que nos interesan:

ORD	ANYO	TOTAL
4	1994D	4404.16
5	1995A	6356.03
6	1995B	7547.14
7	1995C	8457.62
8	1995D	9282.36
9	1996A	12998.8

Figura 51. Resultado del filtro

Ahora es el momento de obtener un modelo. En primer lugar vamos a añadir un nodo "Plot" para ver la curva de crecimiento. Lo conectamos al nodo "Select" (que se llama "Generated"). Editamos el Nodo "Plot" y en "X field" ponemos ORD, en "Y field" TOTAL, y en Overlay Field nada.

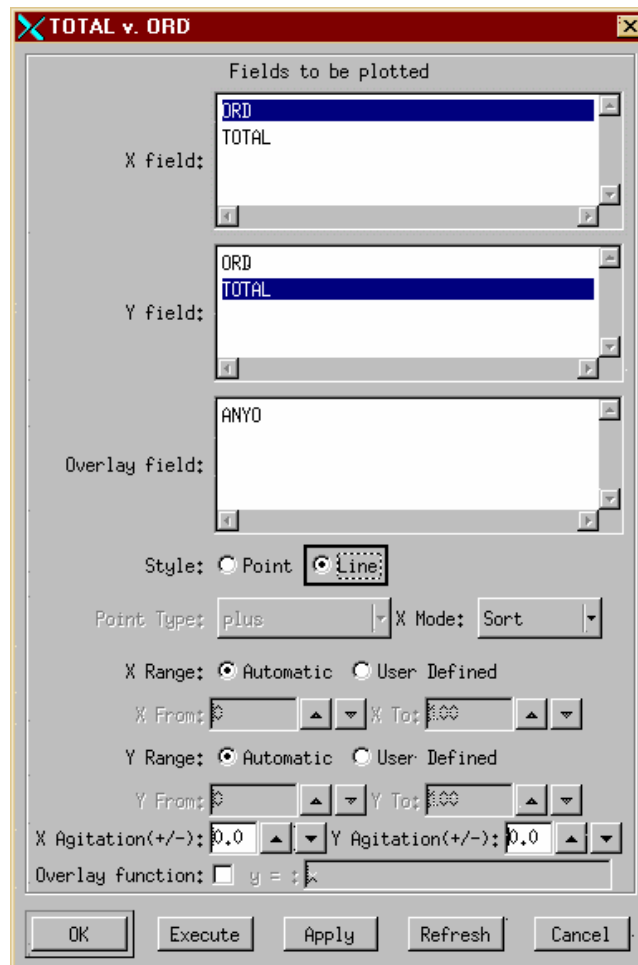


Figura 52. Editando un Plot

El resultado del Plot se ve en la siguiente figura:

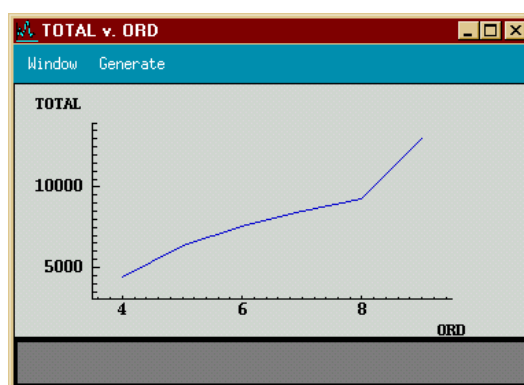


Figura 53. Resultado un Plot

Vemos que se aproxima bastante a la linealidad. Por ello, vamos a aplicar un nodo "Regression" de la categoría "Modelling" y lo conectamos al nodo "Generated".

Ya podemos darle al botón verde de "Execute". Nos aparece un nuevo modelo en la zona derecha del Clementine. Pinchamos dos veces sobre el diamante y nos aparece en la zona de trabajo. Tendremos una situación como la que sigue:

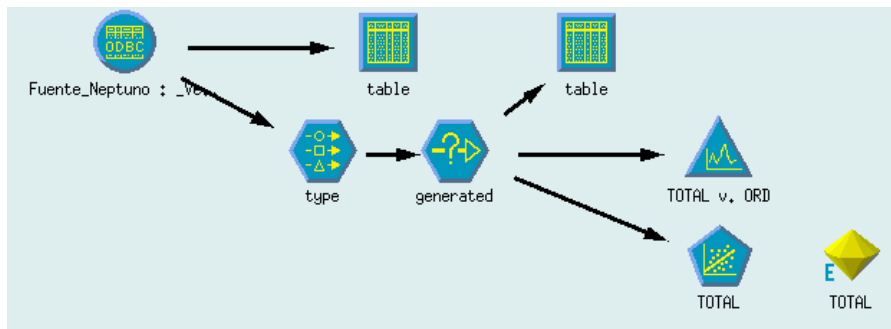


Figura 54. Stream Resultante

Por último, el modelo resultante se puede ver si se selecciona “Browse” en el nodo diamante:

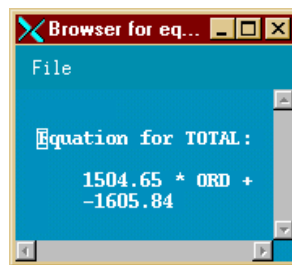


Figura 55. Modelo Resultante (Ecuación lineal)

Para evaluarlo vamos a utilizar el nodo diamante y lo enganchamos a la fuente a través de un nodo type, igual que el anterior. A continuación del nodo diamante “TOTAL” añadimos un nodo “multiplot”, como se muestra en la siguiente figura:

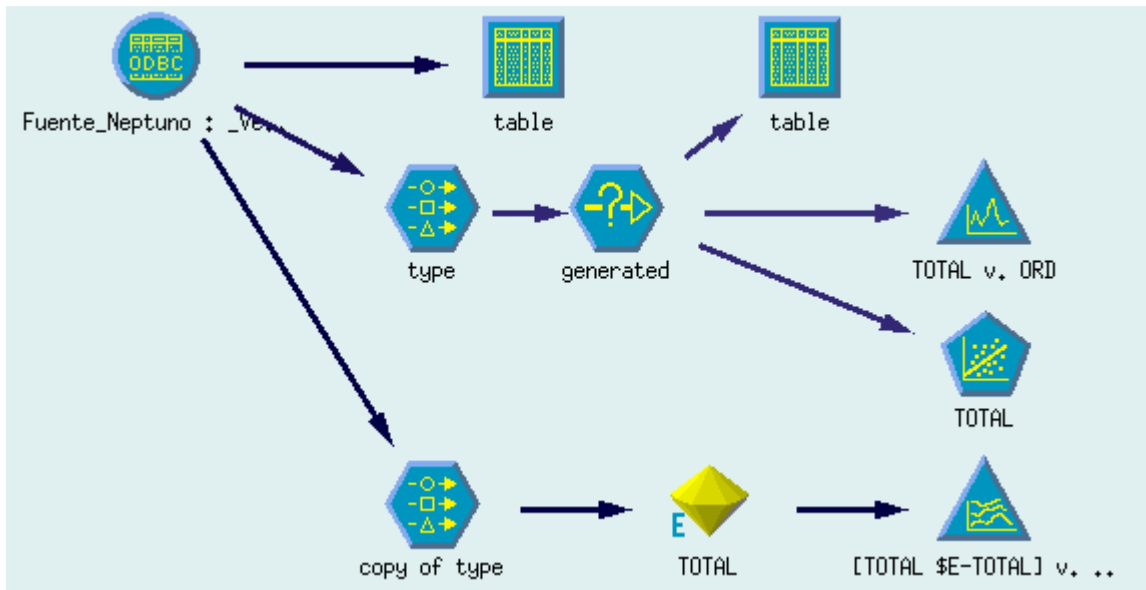


Figura 56. Evaluando el modelo gráficamente

En el multiplot vamos a poner como variable X ORD y como variables Y vamos a poner TOTAL que es el valor real y \$E-TOTAL que es el valor generado.

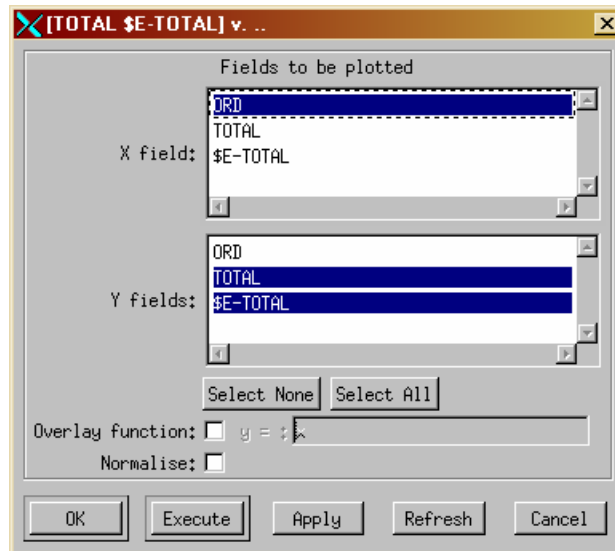


Figura 57. Editando el multiplot

Si lo ejecutamos tenemos el siguiente resultado:

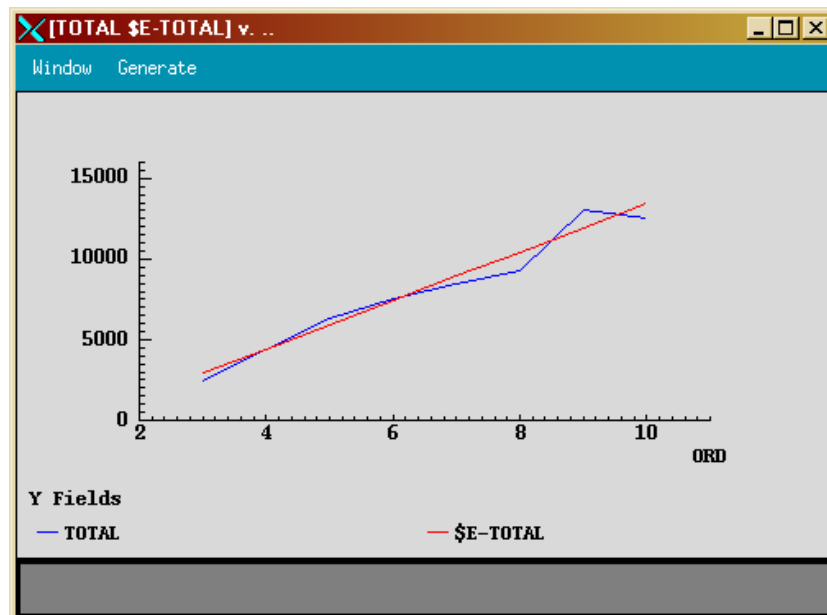


Figura 58. Comparando la curva real con el valor predicho

Donde podemos ver que el modelo se ajusta bastante bien a la curva real.

Ahora si queremos aplicarlo para cualquier valor, utilizamos un nodo "User Input". Para ello cogemos otro nodo diamante y lo enganchamos con el nodo "generated". Pinchamos con el botón derecho en el nodo diamante y allí pinchamos en la opción "User Input Mode", como se ve en la siguiente figura:

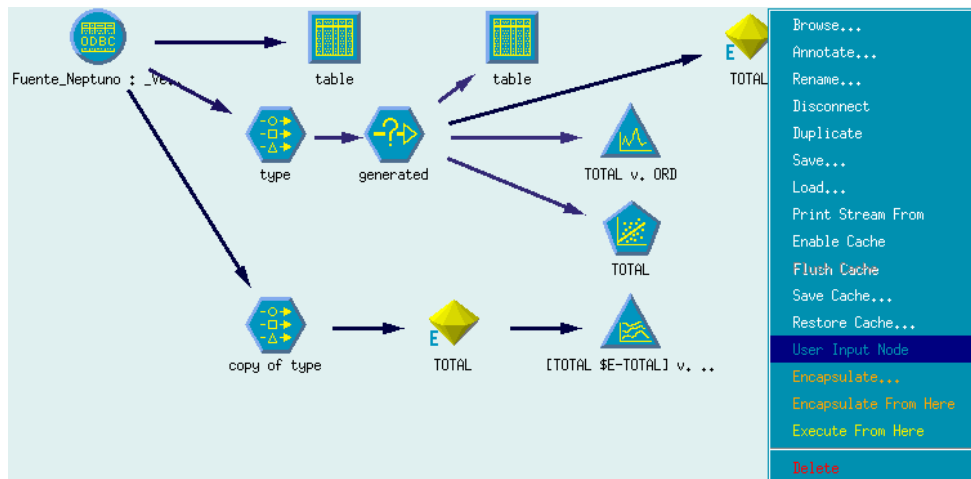


Figura 59. Creando un nodo “User Input Node”

Nos ha aparecido un nuevo nodo “User Input”. Editamos el nodo “User Input” e insertamos los valores que queremos predecir (trimestres 11 y 12) y le damos a OK. Por ejemplo, lo que aparece en la siguiente figura (el 10 también lo ponemos para comparar con lo que teníamos):

Figura 60. Entrada de datos manual

Ahora desconectamos el nodo diamante “Total” del nodo “generated” (Recuerda, pinchando en el enlace con el botón derecho: “Destroy Connection”). Conectamos el nodo “User Input” con el nodo diamante “Total” y éste a su vez con un nuevo nodo Table. Ejecutamos el nodo Table y vemos lo que tenemos en la figura siguiente:

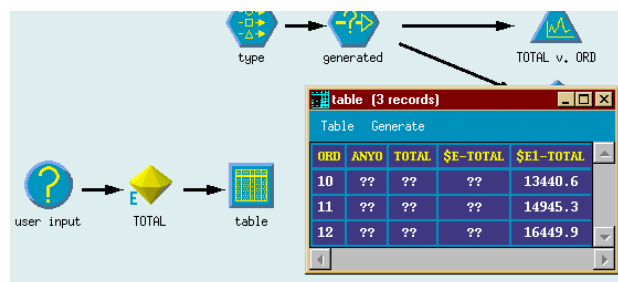


Figura 61. Utilizando el modelo para predecir los datos

Como vemos, predecimos un valor de 13440,6 para el trimestre 10 (1996B), que supusimos no terminado (y que es ligeramente superior al existente, 12495, lo que parece lógico), y unas predicciones de 14945,3 para el trimestre 11 (1996C) y una predicción de 16449.9 para el trimestre 12.

Ahora podemos grabar el stream.

4. Validación de Modelos

Recuperamos el stream de los fármacos que hemos realizado en el apartado 3.1. Vimos que existían cuatro hospitales. Si ejecutamos el nodo "Analysis" enlazado con el nodo diamante "Drug2" teníamos:

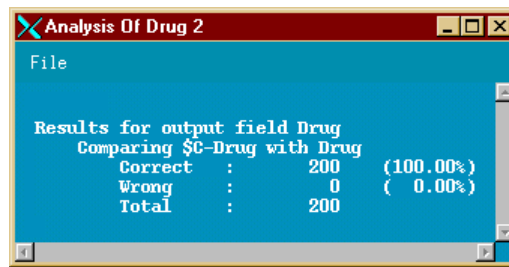


Figura 62. Resultado del modelo Drug2 con el primer hospital

El último modelo obtenido para los fármacos tenía un 100% de precisión con respecto de los datos de entrenamiento. Esto, en principio, parece óptimo, pero no es así.

Obtener un 100% sobre los datos de entrenamiento es relativamente sencillo y no asegura que el modelo se vaya a comportar bien. Para saber si el modelo se va a comportar bien, debemos comprobarlo con datos "frescos", es decir, con datos que no hayan intervenido en el aprendizaje del modelo.

En este problema la manera de actuar parece directa. Si tenemos cuatro hospitales y hemos utilizado el primer hospital para obtener el modelo, podemos comprobar el modelo con el resto de hospitales para ver si funciona bien.

Como nos va hacer falta espacio, primero organizamos un poco el stream de los fármacos en la parte superior de la pantalla, como sigue:

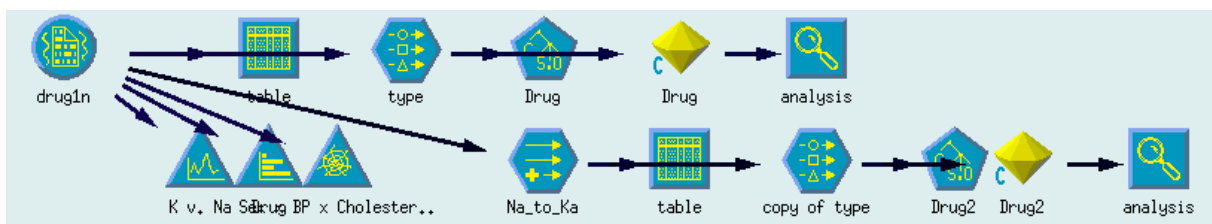


Figura 63. Stream de Fármacos para generar y comprobar el modelo de un hospital

Se trata de duplicar el camino de validación para el resto de hospitales. Para el segundo hospital, añadimos un nuevo nodo "Var. File" y lo enganchamos con el fichero "..\LabKDD\drugs\Drugs2N", marcando el checkbox para que coja el nombre de los atributos del fichero. Ahora, en vez de repetir y configurar todos los nodos que nos hacen falta, pincharemos con el botón derecho en aquellos nodos que nos interesan y seleccionaremos "Duplicate", para hacer copias de tantos nodos como nos hagan falta. En concreto, se trata de construir lo siguiente:

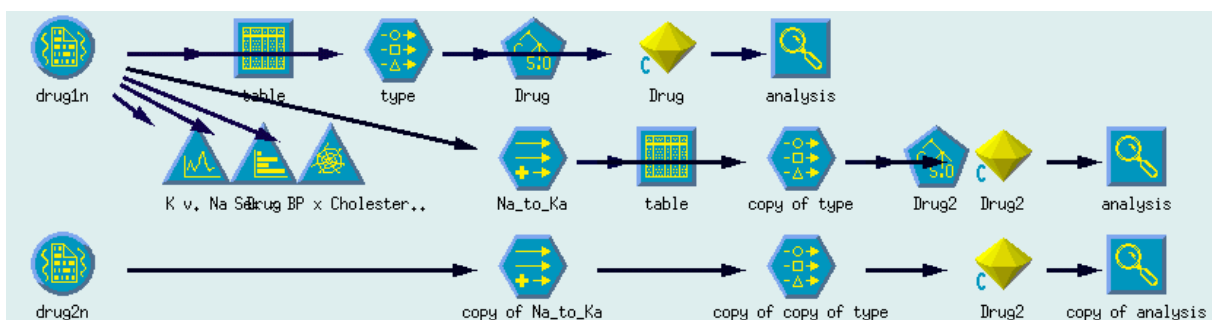


Figura 64. Stream de Fármacos para comprobar el modelo para el segundo hospital

Una vez conectados como se ve en la figura, se trata de ejecutar el último nodo "copy of analysis" para ver qué resultados obtenemos con los 400 registros de drug2n:

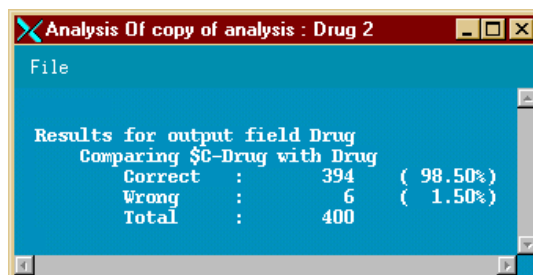


Figura 65. Resultado del modelo Drug2 con el segundo hospital

El resultado, aunque ya no es del 100%, sigue siendo bastante bueno, lo que hace suponer que el modelo "Drug2" es un buen modelo.

Podemos repetir la jugada con el tercer y cuarto hospital, obteniendo:

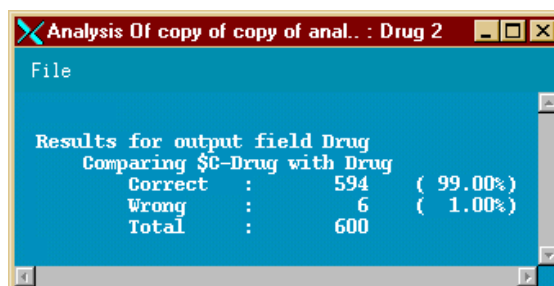


Figura 66. Resultado del modelo Drug2 con el tercer hospital

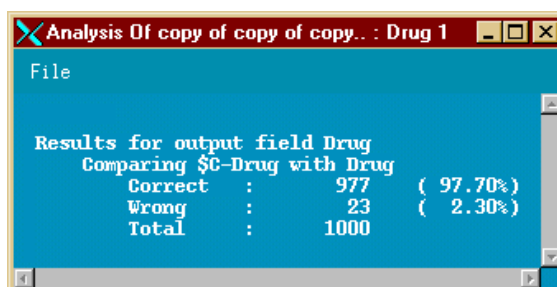


Figura 67. Resultado del modelo Drug2 con el cuarto hospital

Resultados muy aceptables. **Grabamos el stream con otro nombre, p.ej. "drug2.str"**.

4.1.1 Cross-Validation

Aunque el error es pequeño se podría pensar que sólo se han utilizado 200 datos para generar el modelo y 2.000 para validarlo. ¿Existe alguna manera de utilizar gran parte de los datos y aún así tener una referencia para saber lo bueno que es el modelo?

Una herramienta sencilla en esa línea es el método de "Cross-validation".

Para ello, en primer lugar, vamos a fusionar los datos. Desconecta los cuatro nodos de fuente de datos "drug1n", "drug2n", "drug3n" y "drug4n". Añade un nuevo nodo "Append" y engancha las cuatro fuentes con él. Finalmente, engancha un nuevo nodo "Table" con el nodo "Append" y obtendrás lo que se muestra en la siguiente figura:

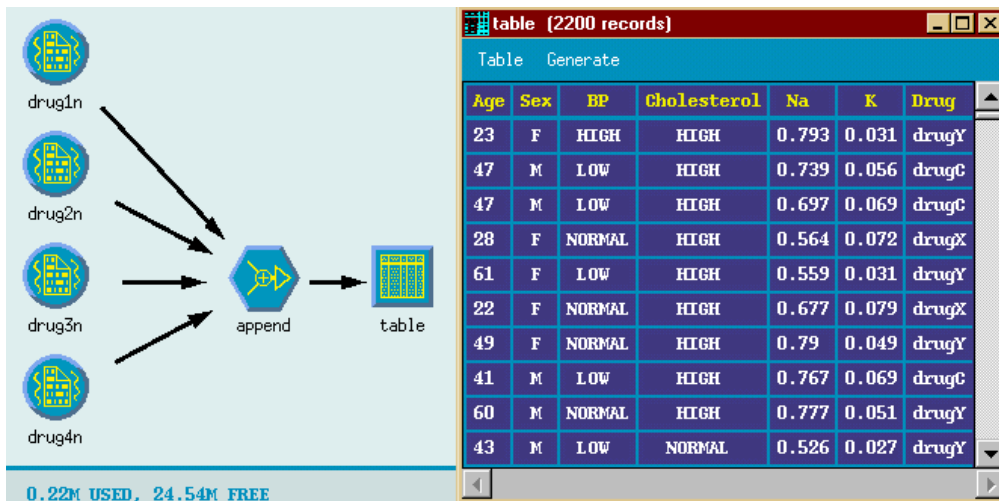


Figura 68. Juntando los datos de los diferentes hospitales

Hemos juntado los 2200 registros de todos los hospitales.

Podemos observar los datos con las mismas herramientas que hicimos, con un plot o con un nodo “Matrix” para comparar el campo BP y el campo Cholesterol. No hay ninguna relación significativa aparte de la que ya vimos, la relación entre Na/K. Añadimos de nuevo el atributo derivado Na_to_K, añadimos un nodo “Type” para marcar el campo Drug como salida y añadimos un nodo “Build C5.0”. Lo conectamos todo y tenemos lo que se muestra en la siguiente figura:

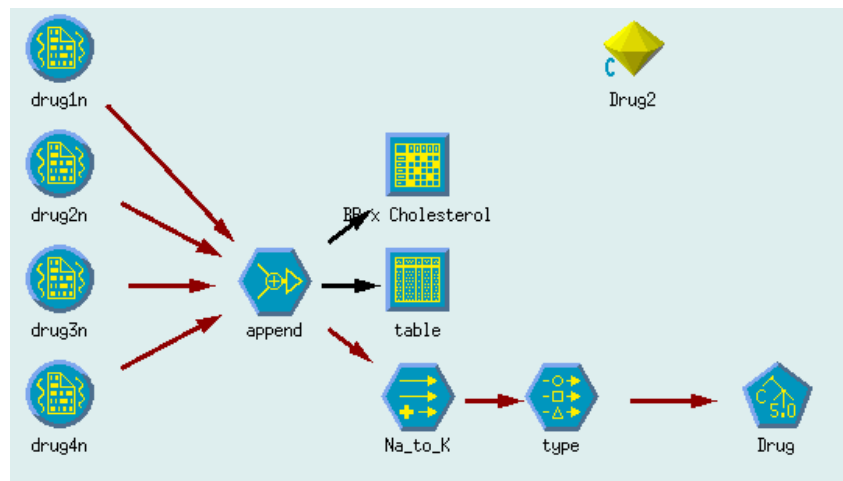


Figura 69. Obteniendo un stream con todos los datos (El nodo Drug2 es el del modelo anterior)

Ahora, en las propiedades del nodo “C.5.0 Drug”, vamos a marcar “Cross-Validation”, como se muestra en la siguiente figura:

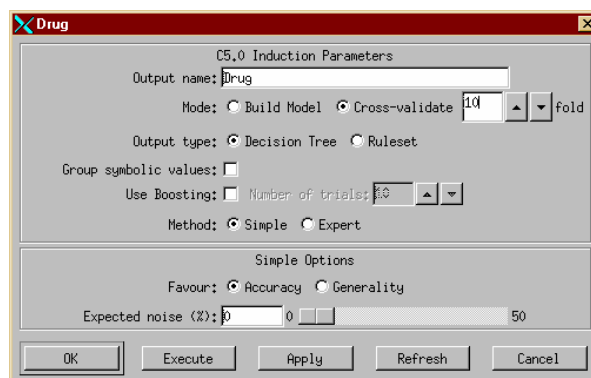


Figura 70. Activando Cross-validation

Se va a generar el modelo con 10 particiones disjuntas de los datos y probarlo con el resto para ver cómo se comporta. Esto nos dará una estimación de cómo se comportaría con nuevos datos. De hecho, si ejecutamos ahora no nos genera ningún modelo, sino que nos muestra la siguiente pantalla de información de Cross-validation:

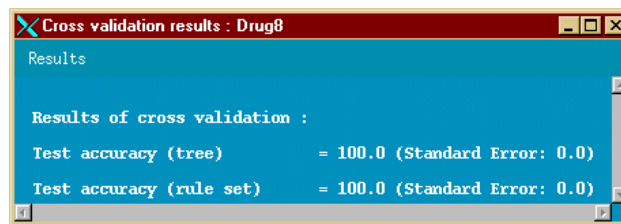


Figura 71. Resultado de Cross-validation

Un 100%! No se obtienen esos datos todos los días... Eso quiere decir que el modelo que se podría generar con todos los datos es extremadamente fiable.

Pues generémoslo. Para eso añadimos un nuevo nodo "Build C5.0" que ahora sí va a generar modelo. Llamémosle "Drug4" (ver Figura 80). Generemos el modelo y comparemos (Browse) el modelo viejo con el nuevo. En la siguiente figura, se ve la diferencia sutil entre los dos modelos:

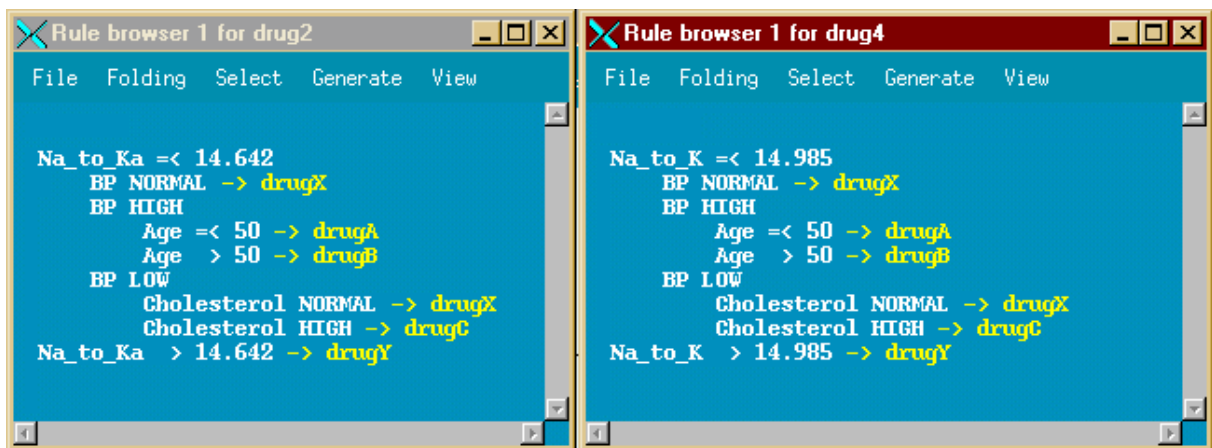


Figura 72. Comparación del modelo con 200 datos y el modelo con 2200

4.1.2 Sampling & Boosting

En el caso anterior los resultados eran tan buenos que no se podía apreciar nítidamente la función del cross-validation. Vamos a empeorar el problema mediante la reducción drástica del número de ejemplos. Para ello introducimos un nuevo nodo "Sample" de la categoría "Record Ops" y lo enganchamos justo detrás del nodo "type". Editamos el nodo "Sample" para quedarnos sólo con 25 ejemplos!



Figura 73. Tipo de Muestreo (25 ejemplos máximo, con un 10% para generar el muestreo)

Ahora, si al resultado del muestreo (25 registros) añadimos un nodo "Build C5.0" que calcule la validación cruzada tenemos lo siguiente (ten en cuenta que al ser un muestreo aleatorio tus datos pueden cambiar):

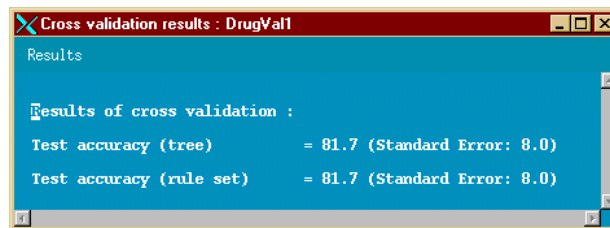


Figura 74. Resultado de Cross-validation para 10-fold cross-validation

Vemos que la accuracy que se puede esperar ahora es mucho menor (81,7%) con un intervalo de $\pm 8.0\%$.

Ahora vamos a aplicar un modelo "Build C5.0" sobre los sólo 25 registros. Si lo aplicamos y lo contrastamos con los 2.200 registros, el resultado es un modelo pobre, como podemos ver en la siguientes figuras:

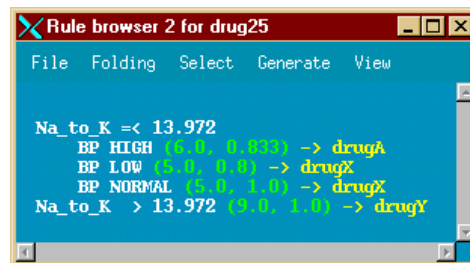


Figura 75. Modelo generado para 25 registros

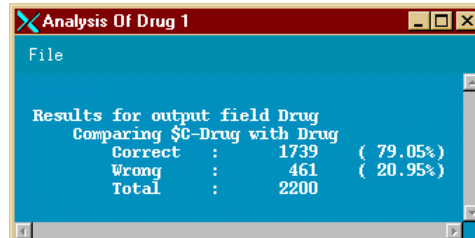


Figura 76. Precisión del modelo con 25 ejemplos

Lo cual se ajusta bastante a los valores que nos daba la validación cruzada.

De todas manera, aunque dispongamos de pocos datos, podemos hacerlo mejor.

Existe una técnica llamada "Boosting", que realiza varios modelos para distintos subconjuntos de los datos y luego combina esos modelos. Para verlo, vamos a aplicar de nuevo un modelo "Build C5.0" sobre los 25 registros, pero ahora activando la opción "Boosting":

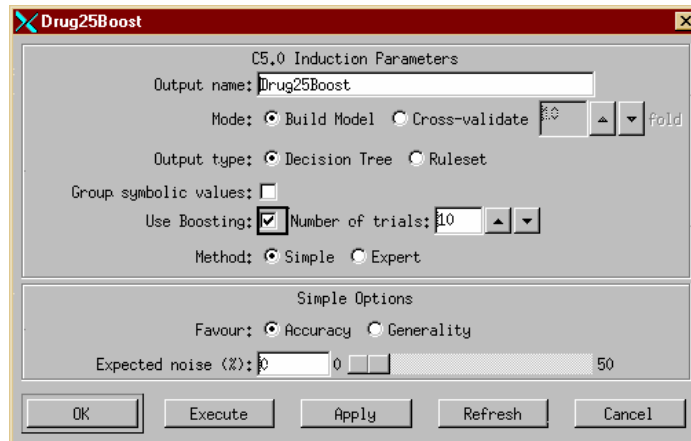


Figura 77. Activando el BOOSTING

Si lo aplicamos y lo contrastamos con los 2.200 registros (antes del muestreo), el resultado es un modelo ligeramente mejor (o mejor dicho una combinación de modelos, de hecho el primer valor de cobertura ahora es decimal), como podemos ver en las siguientes figuras:

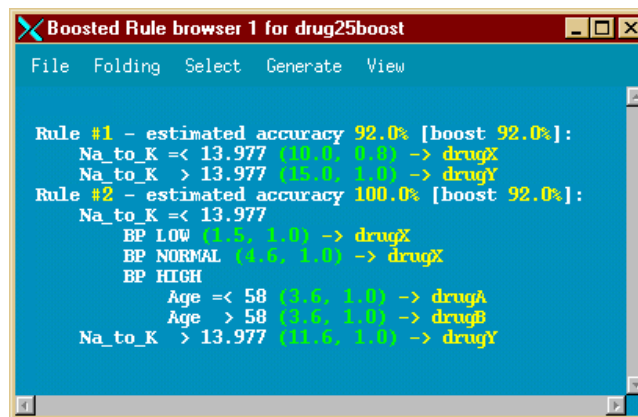


Figura 78. Modelo generado para 25 registros con BOOSTING

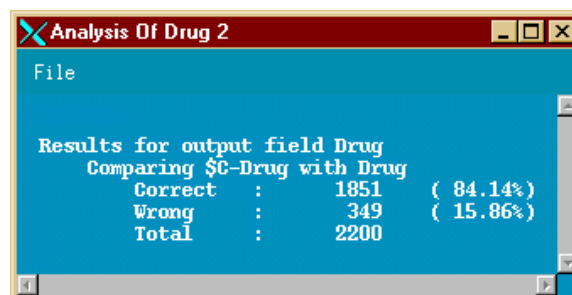


Figura 79. Precisión del modelo con 25 ejemplos con BOOSTING

Con unos resultados ligeramente mejores. El grafo de streams final es el siguiente:

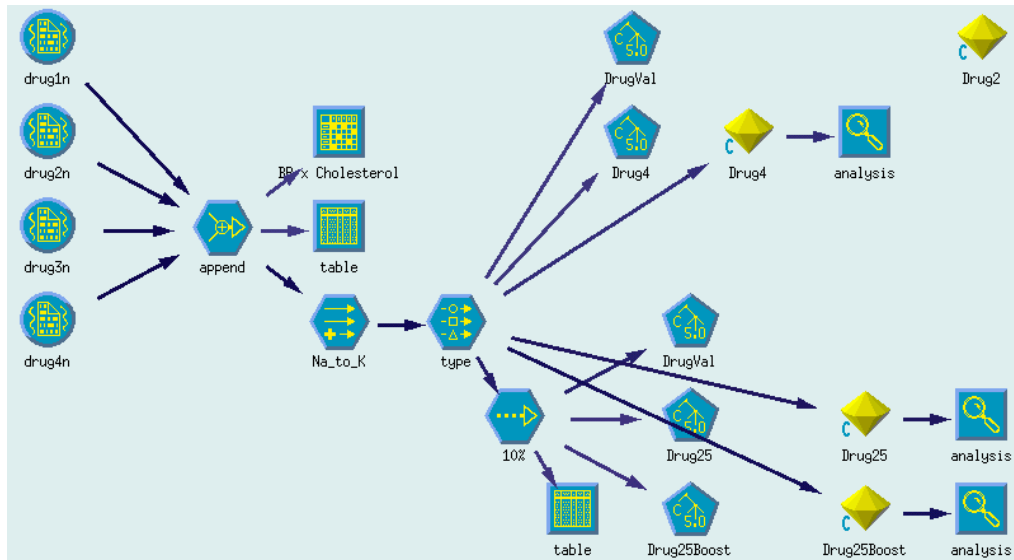


Figura 80. Grafo de Streams

Graba el stream con otro nombre.

CUESTIÓN: ¿Qué hemos aprendido acerca del tamaño de muestra, la partición de la misma y la generación de muchos modelos?

SEGUNDA PARTE

En esta segunda parte, los ejercicios serán mucho menos guiados y se darán algunas pistas para su resolución. En algunos casos se pondrán los streams finales. Además, el tamaño y complejidad de los datos es mayor en esta segunda parte.

5. Patrones Predictivos

Vamos a analizar algunos problemas que requieren la extracción de patrones predictivos. De hecho, la mayoría de los problemas que hemos visto en secciones anteriores son patrones predictivos.

5.1 Patrones Secuenciales

Vamos a abordar un problema de predicción secuencial. En el directorio “..\LabKDD\camara” tienes 2.422 datos de tomas sobre una cámara refrigeradora. Los atributos son:

`Time, Power, Temperature, Pressure, Uptime, Status, Outcome`

Que representan el tiempo, la potencia que requiere la máquina, la temperatura, la presión, un valor interno de funcionamiento, el estado y el tipo de tratamiento. La cámara soporta cuatro tipos de tratamiento (0, 101, 202, 303 y 404).

Se pretende predecir la potencia según el tiempo y el resto de las variables. Para ello se sugiere utilizar regresión y, si se desea, redes neuronales.

En primer lugar, si conectamos el nodo fuente y vemos la evolución de la potencia a lo largo del tiempo (mediante un plot):



Figura 81. Atacando el problema de las Cámaras

vemos que existen muchos casos diferenciados:

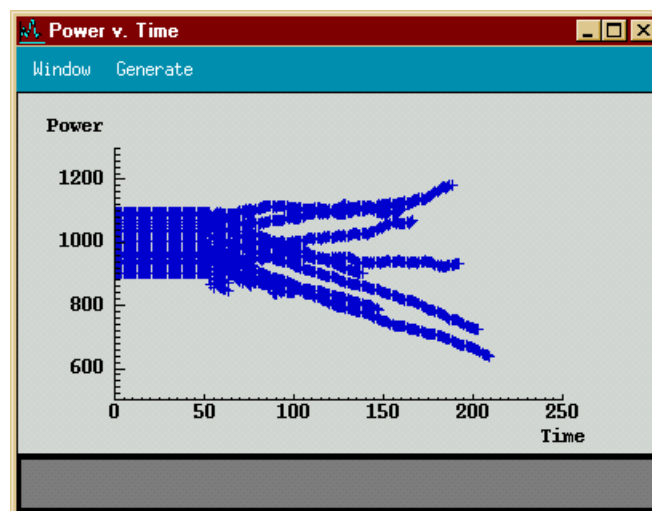


Figura 82. Plot de la Potencia según Tiempo

Vemos que la potencia no sólo depende del tiempo, aunque, aparentemente, si se fija “la serie”, aparecen curvas de tendencias casi lineales, con lo que, aparentemente, si se consiguiera separar correctamente los datos, puede ser que la regresión lineal a partir del tiempo únicamente funcione bien.

Se propone pensar en atacar el problema directamente con una red neuronal, o bien partir el problema según algún atributo, que permita a posteriori aplicar regresión lineal. En caso afirmativo compara los dos resultados. ¿facilita el Clementine la combinación de modelos?

Si la cámara sólo permitiera reguladores lineales, ¿qué modelo deberías utilizar?

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 85. Se recomienda intentarlo al menos 10 minutos antes de echar mano del mismo.

5.2 Clasificación

Aunque de este tipo hemos hecho el mayor número de problemas, vamos a realizar dos ejemplos utilizando varios métodos, en particular: árboles de decisión (C5.0), redes neuronales (Trainnet) y reglas (Build Rule) (v5.2) / C&R trees (v.6.0).

5.2.1 Varios modelos para los medicamentos

Aplica todos ellos al ejemplo de los medicamentos y examina qué método da mejores resultados. Como hay suficientes datos, realiza una partición de 1.700 para entrenamiento y 500 para test. Los 500 de test que sean iguales para todos los modelos.

Para ello se recomienda juntar todas las fuentes (append) y después separarlas mediante un nodo sample que incluya los primeros 1.700 (pass sample) y otro que excluya los primeros 1.700 (discard sample), quedándose con los 500 restantes. Evalúa los resultados.

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 86.

El problema de la partición anterior es que no es aleatoria. Desgraciadamente el Clementine no permite hacerlo directamente. Pero sí que lo podemos hacer indirectamente. Para ello en el directorio de los ejemplos se ha dejado un fichero "random-column.xls" que contiene una columna de 40.000 números aleatorios. Además hay un fichero "random-column.txt" generado a partir del fichero xls que puede ser incorporado como fuente en el Clementine. Si nos quedamos con los 2.200 primeros podemos añadir esta columna (mediante el nodo merge) a los 2.200 datos provenientes de los medicamentos. Después de esto sólo tendremos que ordenarlos por el valor aleatorio antes de hacer la partición. Con esto conseguiremos un muestreo realmente aleatorio sobre los datos de drugs

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 87.

5.2.2 Varios modelos para el monks1

En el directorio "..\LabKDD\monks1" tienes tres ficheros: "monks1.data", "monks1.names" y "monks1.test". Se trata de un problema ficticio del repositorio UCI con cinco atributos discretos y una clase booleana (al tipar los atributos los puedes coger como autosymbol). El fichero "monks1.names" es una descripción del problema, el fichero "monk1.data" se utilizará como datos de entrenamiento y el fichero "monks1.test" como datos de test/validación.

Intenta obtener modelos predictivos para las clases utilizando los tres tipos de modelos de clasificación que dispone el Clementine. Compara los resultados.

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 90. Se recomienda intentarlo al menos 10 minutos antes de echar mano del mismo.

6. Patrones Descriptivos

6.1 Reglas de Asociación y Determinación

Vamos a tratar ahora el típico caso de la cesta de la compra. En el directorio "..\LabKDD\cesta" tienes 1000 cestas de la compra con los siguientes atributos:

`cardid, value, pmethod, sex, homeown, income, age,`

que significa el identificador de la tarjeta, el valor de la compra, el método de pago, el sexo del comprador, si es propietario de su casa, sus ingresos y su edad. Además se indica para cada cesta si ha comprado de estas categorías:

```
fruitveg, freshmeat, dairy, cannedveg, cannedmeat, frozenmeal,
beer, wine, softdrink, fish, confectionery
```

Vamos a intentar determinar las posibles asociaciones entre productos. Para ello utiliza el nodo gráfico "Web" y el nodo modelo "A priori". Para éstos debes tipar que todos los atributos de compra son entrada y salida (BOTH) y que los atributos del comprador son irrelevantes (NONE).

Una vez enganchados y ejecutados, notarás que el modelo "a priori", para poderlo ver se ha de pinchar con el botón derecho en la parte derecha del Clementine, porque este tipo de modelos no se pueden arrastrar al área de trabajo. El resultado al pulsar "Browse" sobre él debe ser similar al siguiente:

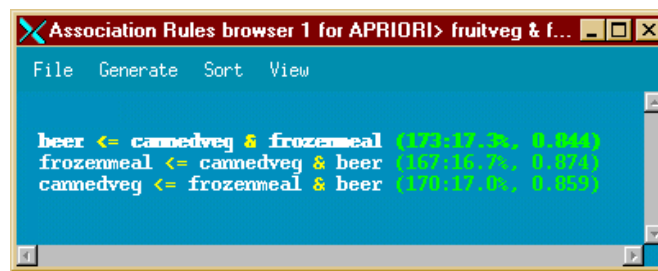


Figura 83. Reglas de asociación más relevantes generadas por Apriori.

El primer valor indicado es el *soporte* y el segundo la *confianza*. P.ej. para la primera se dice que es aplicable a 173 instancias (lo que supone un 17,3% de los datos) y que en éstas se cumple en 84,4% de las veces.

Una vez determinadas estas asociaciones (visto también el gráfico web), queremos ver si hay algún determinado tipo de compras que determine el sexo del comprador.

Para ello vamos a utilizar un nodo GRI, indicando como atributos de entrada todos y como atributo de salida el sexo. Cuando lo generamos, pasa lo mismo que con el nodo "A priori", sólo se puede ver como "Browse" desde el área de la derecha del Clementine.

El resultado al pulsar "Browse" sobre él debe ser similar al siguiente:



Figura 84. Reglas de determinación para el sexo más relevantes generadas por GRI

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 91. Se recomienda intentarlo al menos 10 minutos sin echar mano del mismo.

Vuelve a realizar el stream utilizando dos atributos derivados flag, mediante el nodo "SetToFlag" para que te genere un atributo SexM y SexF que sean opuestos. Elimina el atributo Sex y vuelve a generar los modelos. Compara los resultados con el caso anterior.

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 92.

6.2 Estudios de Correlación

Se trata de utilizar el nodo Statistics, que permite realizar correlaciones entre atributos, además de otras estadísticas.

Estudia la correlación entre los distintos atributos numéricos del problema de la cámara. ¿Encuentras alguna correlación significativa? ¿Puede ayudarte a mejorar los resultados que obtuviste anteriormente?

6.3 Segmentación

Se trata de utilizar los nodos Train Kohonen, K-means.

Retomemos de nuevo el problema de los fármacos. Se trata ahora de segmentar los pacientes, según tipología. Para ello, vamos a utilizar los datos del primer hospital para entrenamiento y los datos del cuarto hospital para evaluación.

Genera un stream en el que generes el atributo derivado K_to_Na y luego excluyas los atributos K y Na (al ya tener el atributo derivado). Tipa todos los atributos que te quedan como IN, exceptuando el drug, que lo excluiremos también poniéndolo como OUT.

Genera un modelo Train Kohonen y otro K-means. Ejecútalos.

Aplica esos modelos al cuarto hospital (duplicando los nodos de proceso que te hagan falta). Examina ahora los resultados de ambos modelos y examina las particiones que generan. ¿Te parecen lógicas?

El stream propuesto para esta actividad se encuentra en el apéndice, Figura 93. Se recomienda intentarlo al menos 10 minutos sin echar mano del mismo.

7. Generación y Exportación de Resultados

Se trata de utilizar la exportación a Excel, los nodos File, Report, ODBC Output, etc. Pregunta a tu profesor para que te proponga un ejercicio.

8. Combinación y Ponderación de Modelos

Ejercicios:

1. ¿Cómo combinarías los resultados del ejercicio de los fármacos con varios modelos (red neuronal, árbol C5.0, reglas) para obtener un modelo más potente? ¿Proporciona Clementine una solución para este problema?
2. Imagina que tienes la siguiente matriz de costes para los medicamentos:

	DrugA	DrugB	DrugC	DrugX	DrugY
Adm. Correctamente	1000€	2000€	500€	1000€	1500€
Adm. Incorrectamente	1500€	3500€	600€	10000€	3000€

Intenta ver cómo obtener un modelo que minimice costes. Para ello construye una matriz de confusión para cada modelo. Luego multiplica la matriz de costes por la de confusión y mira qué matriz resultante es mejor.

¿Proporciona Clementine una solución para automatizar este problema?

TERCERA PARTE

En esta tercera parte, sólo se plantea el problema y se deja completa libertad para extraer modelos, es decir para hacer una minería de datos autónoma. El tamaño y complejidad de los datos en esta tercera parte es mucho mayor, de hecho algunos de los problemas son concursos de conferencias del campo, con lo que competiremos a un alto nivel!

9. Ejercicios Libres

Los ejercicios libres son los siguientes y se encuentran en el directorio (no hace falta bajárselos de la página web). Algunos son tan libres que simplemente se trata de sacar cuantos más patrones útiles mejor (no hay tareas predeterminadas).

9.1 *Tamaño Pequeño/Medio:*

UCI (VARIOS)

Casi todos son de carácter predictivo, aunque se pueden usar para otros fines.

<http://www.ics.uci.edu/pub/machine-learning-databases/>

ML-DATASETS (VARIOS)

<http://www.recursive-partitioning.com/mv.html>

9.2 *Gran Tamaño*

Para trabajar con datos de gran tamaño es posible que necesites cambiar la memoria disponible del Clementine. Esto se realiza en el menú Options → Memory Limit.

SWISSLIFE DATASET (Ámbito: financiero):

<http://research.swisslife.ch/kdd-sisyphus/>

PKDD'99 Discovery Challenge (Financial domain and Medical domain)

<http://lisp.vse.cz/pkdd99/Challenge/chall.htm>

KDD Cup 2000

<http://www.ecn.purdue.edu/KDDCUP/>

UserName: kddcup

Password: legcare4KDD

The COIL Challenge 2000 (The Insurance Company)

<http://www.liacs.nl/~putten/library/cc2000/>

PKDD 2001 Challenges

http://www.informatik.uni-freiburg.de/~ml/ecmlpkdd/discovery_challenges.html

The Predictive Toxicology Challenge for 2000-2001 (MUY COMPLEJO)

The Discovery Challenge on Thrombosis Data (Thrombosis)

KDD 2001-05-09

<http://www.cs.wisc.edu/~dpage/kddcup2001/>

Todavía no disponible

APÉNDICE

En este apéndice se muestran los streams “solución” de los ejercicios de la parte 2.

10. Soluciones

10.1 Problema de la Cámara:

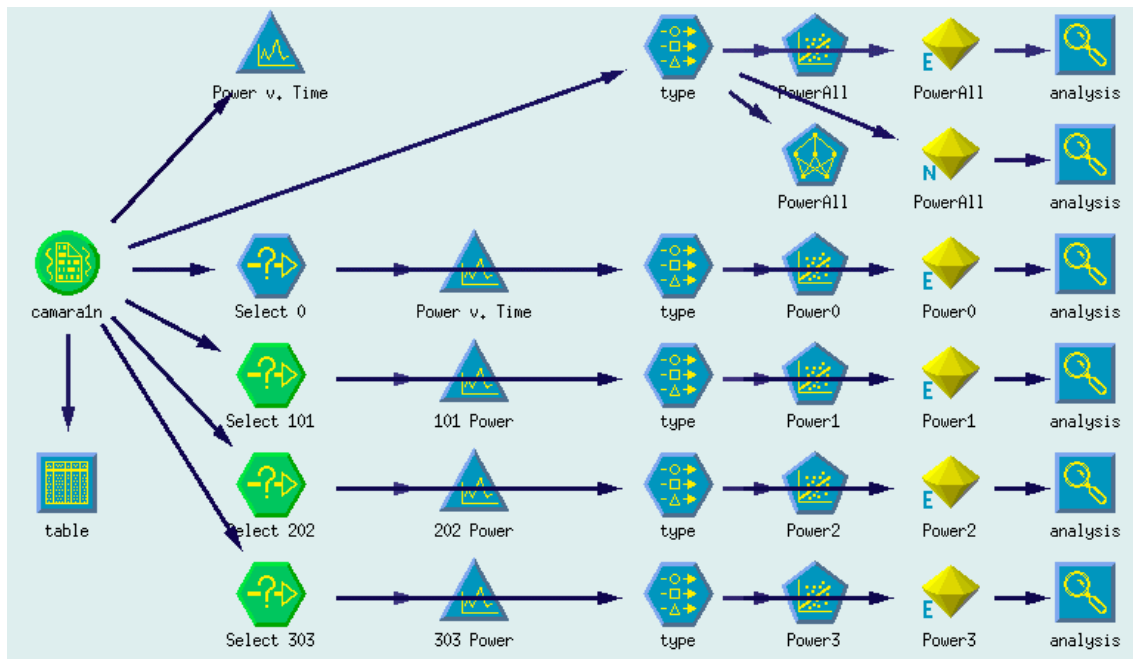


Figura 85. Stream de la Cámara. Se observa la red neuronal sobre todos los datos arriba y la regresión partida por el campo “Outcome”.

10.2 Problema del muestreo de fuentes de drugs (1700 – 500):

10.2.1 Muestreo no aleatorio:

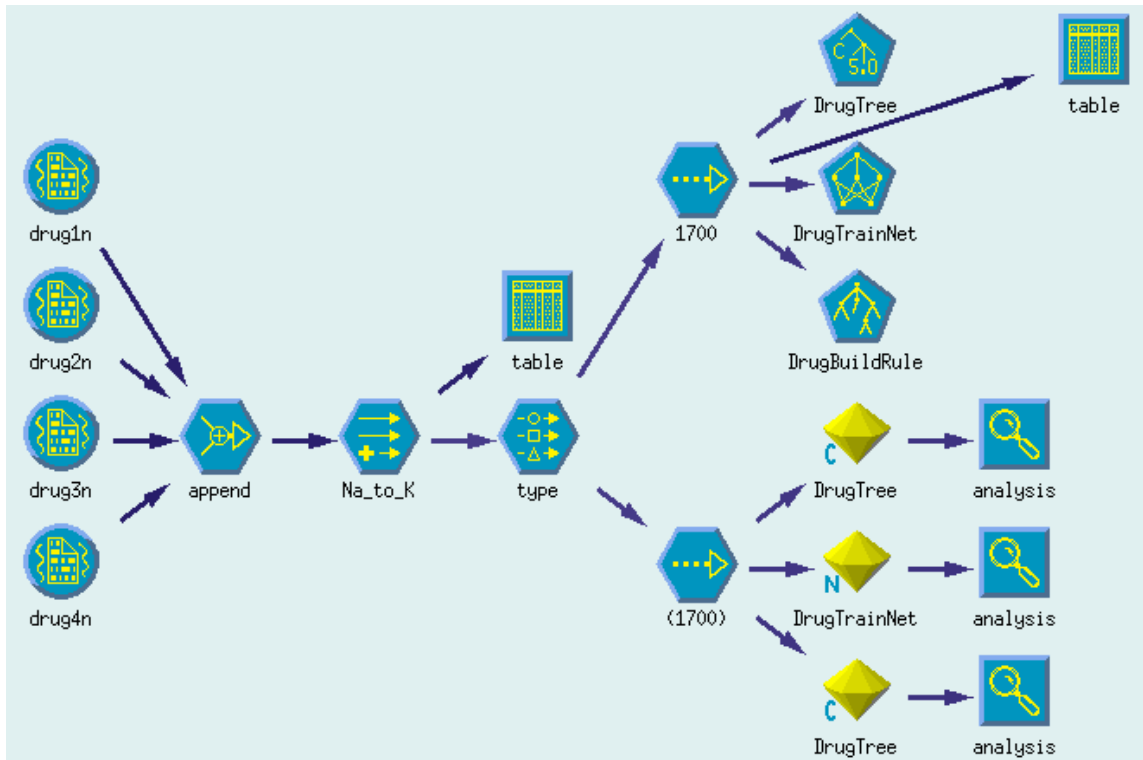


Figura 86. Stream para hacer un muestreo sobre los datos de drugs. Se juntan todas las fuentes (append) y después se separan mediante un nodo sample que incluye los primeros 1700 (pass sample) y otro que excluye los primeros 1700 (discard sample), quedándose con los 500 restantes.

10.2.2 Muestreo aleatorio:

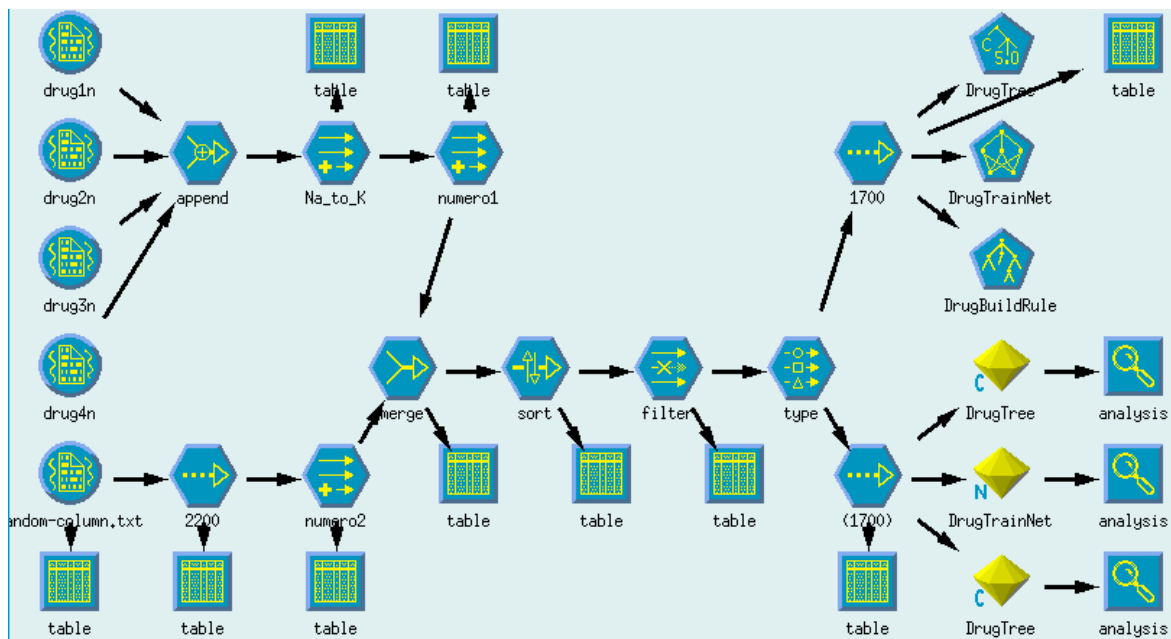


Figura 87. Stream para hacer un muestreo realmente aleatorio sobre los datos de drugs. Para ello se añade una columna proveniente del fichero random-column.txt y después se ordena por este valor aleatorio, procediéndose igual que antes.

Las siguientes figuras muestran la configuración de algunos nodos:

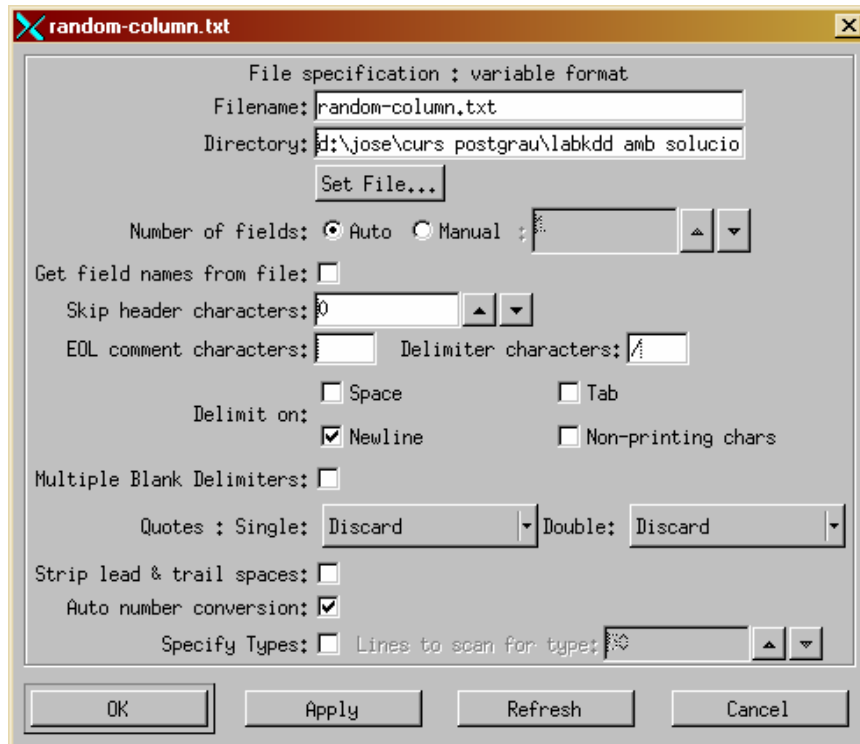


Figura 88. El fichero a incluir contiene reales separados por comas (notación española) y eso marea al convertidor. Para evitar problemas se puede cambiar el Delimiter carácter.

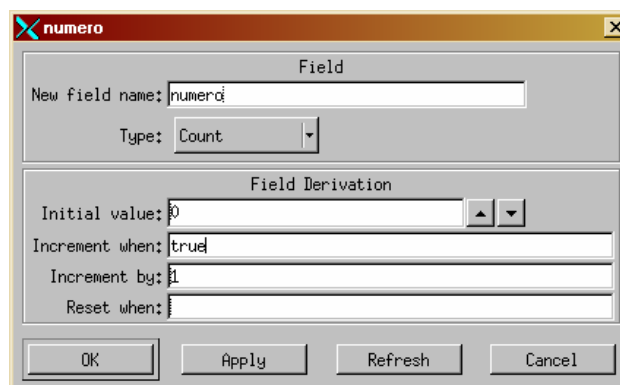


Figura 89. Manera de generar un campo autonumerado.

10.3 Problema del Monks1:

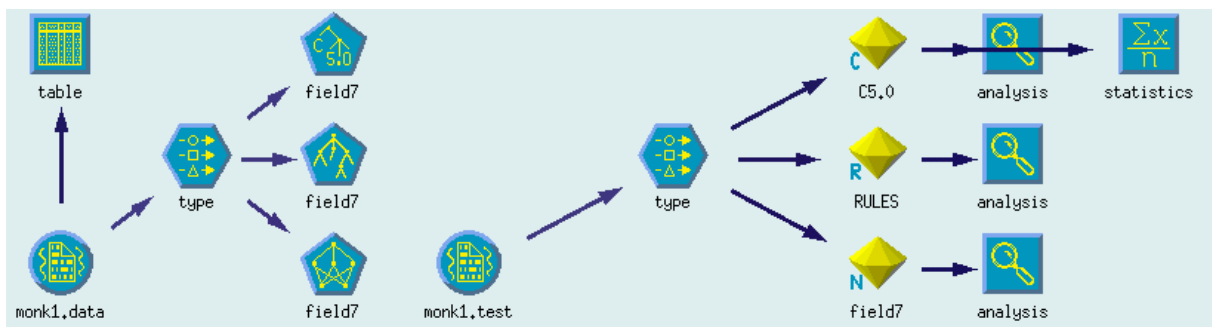


Figura 90. Stream del Monks1.

10.4 Problema de la Cesta de la Compra:

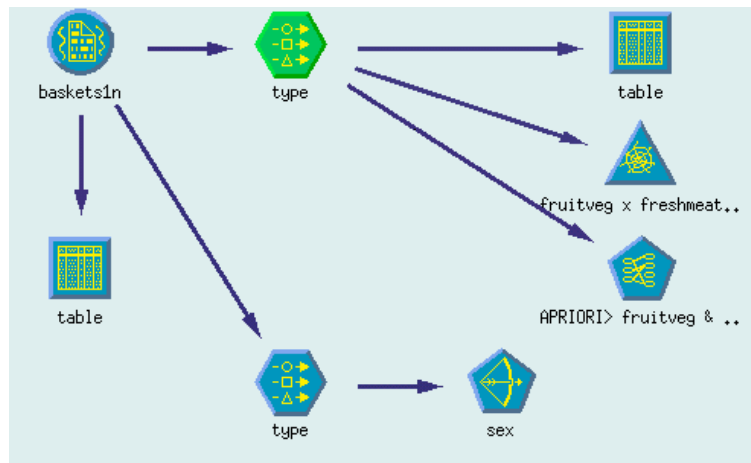


Figura 91. Stream de la Cesta de la Compra.

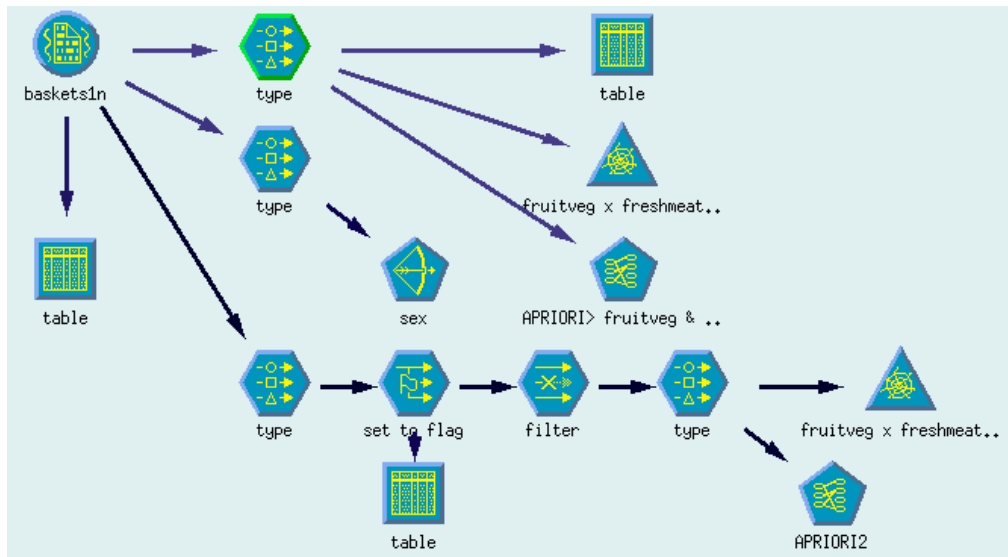


Figura 92. Stream de la Cesta de la Compra generando atributos derivados flag para el sexo.

10.5 Problema de segmentación de pacientes:

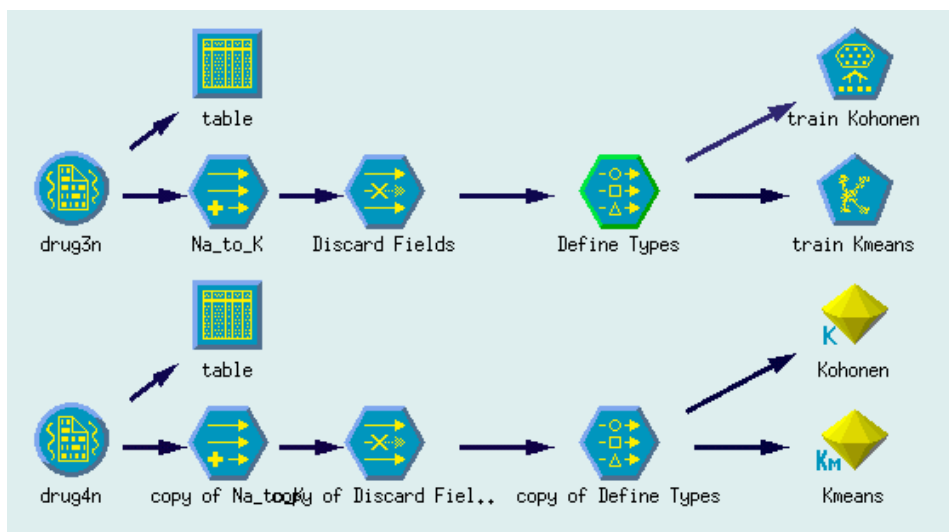


Figura 93. Segmentación del problema de los fármacos.