Universidad de las Ciencias Informáticas
Facultad 4

Título: Propuesta de un Mecanismo de Visualización de los Resultados del Análisis Inteligente de Datos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(s): Anisley Caridad Rodriguez Ferrera

Ricardo Romero Chinea

Tutor(s): Lic. Eddy Manuel Infante Alonso

Ing. Julio Cesar Días Vera

Julio, 2007

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Anisley Caridad Rodriguez Ferrera

_____

Julio Cesar Diaz Vera

_____

Ricardo Romero Chinea

_____

Eddy Manuel Infante Alonso

_____

Datos de Contacto

Pensamiento

"Nada sugiere tanta y tan hermosa literatura como un párrafo de ciencia"

José Martí

AGRADECIMIENTOS

Queremos agradecer:

A la Revolución por habernos dado la oportunidad de formar parte de un proyecto de tal magnitud y permitirnos realizarnos como profesionales, por permitir que nuestros sueños se hicieran realidad.

Principalmente a nuestros padres, hermanos, tíos y abuelos que sin ellos no seríamos lo que hoy somos y no hubiéramos llegado hasta aquí. Gracias a su apoyo incondicional y por confiar siempre en nosotros. A Reinaldo y Din por su preocupación y apoyo.

A Yuniesky y Alexander por su valiosa ayuda y oportunas recomendaciones que demostraron ser amigos en cualquier circunstancia.

A todas aquellas personas que de una forma u otra formaron parte de nuestro desarrollo como futuros profesionales. Gracias para los que aportaron un granito de arena en la realización de nuestro proyecto de tesis.

A nuestros tutores Eddy y Julio por su valiosa colaboración

A nuestros amigos Tailys, Yasnaya, Pepe, Yilena, Pupo, Yosdani, Miguel y Jorge por su apoyo y amistad.

Anita y Ricardo

DEDICATORIA

Dedico este trabajo al esfuerzo de mi mamá

Al cariño de me hermana

A mi sobrino

A la memoria de mi papá

Con todo el amor del mundo…

Anita

Dedico este trabajo

Al esfuerzo y dedicación de mi mamá

A mi papá

A mi hermana

A Reinaldito y Reinier

Con todo mi amor y cariño

Ricardo

Resumen

La Universidad de Ciencias Informáticas (UCI) está inmersa en el proceso de automatización de la sociedad cubana y en el desarrollo de estos proyectos se maneja un volumen de información que hacen prohibitivo su análisis de manera tradicional, producto de su complejidad, disponibilidad y las capacidades físicas limitadas de los seres humanos. Además en varios de los proyectos de exportación se prevé la confección de herramientas de soporte a las decisiones gerenciales para lo cual es indispensable contar con un método para llevar a los gerentes la información transformada en conocimiento que puedan usar en su trabajo y presentarla de manera tal que sea útil para los responsables de utilizar esta información, lo que usualmente implica transformarla al estándar que manejan los gerentes, como son las gráficas, los modelos visuales, las relaciones de dependencias etc.

No existe en la UCI una herramienta que permita realizar de manera eficaz este tipo de tarea. Es por eso que este trabajo tiene como objetivo proponer un mecanismo de visualización de información capaz de presentar los resultados del análisis inteligente de datos.

La estrategia de investigación que se utilizó de acuerdo al objetivo de este trabajo es la investigación exploratoria. Dentro de los métodos a utilizar están los de carácter teórico que incluye el método Analítico sintético y los de carácter empírico dentro de este la observación científica.

PALABRAS CLAVE

Técnicas de visualización, PMML, SVG, XSLT, herramienta de visualización.

Tabla de Contenidos

INTRODUCCIÓN

Actualmente se vive en un mundo globalizado, donde las distancias geográficas no existen y el éxito de una empresa depende de su visión para sacar partido de la información que se encuentra a su disposición. Esta situación ha provocado que los trabajos orientados al análisis de los datos, como factor clave para la toma de decisiones en la empresa, se hayan disparado en los últimos años, y se conviertan en el eje conceptual sobre el que gravitan los sistemas de información actuales. Los SI son el conjunto de funciones o componentes interrelacionados. Tienen como objetivo obtener, procesar, almacenar y distribuir información para apoyar la toma de decisiones necesarias para desempeñar las funciones y procesos de negocios de la organización.

Dentro de las cuatro actividades básicas que realiza un SI encontramos: entrada, almacenamiento, procesamiento y salida de información esta última constituye el eje central de la atención de los usuarios finales debido a que ellos precisan utilizarla para validar sus decisiones. Esta tarea se convierte en uno de los problemas abiertos más importantes en el área de análisis de la información, cuyos esfuerzos principales se centran en la visualización de la información debido a su potencial para ayudar a las personas a encontrar la información que ellos necesitan más eficazmente e intuitivamente mediante técnicas de representación del conocimiento, siguiendo el refrán popular “una imagen vale más que mil palabras”.

Los aportes más importantes en este sentido están orientados a desarrollar herramientas capaces de capturar la información, en forma de conocimiento, almacenada en cualquier tipo de repositorio y obtenida desde cualquier vía, ya sea mediante Data Warehouse o utilizando técnicas de Minería de datos etc. y presentarla de manera tal que sea útil para los responsables de utilizar esta información, lo que usualmente implica transformarla al estándar que manejan los gerentes, como son las gráficas, los modelos visuales, las relaciones de dependencias etc.

Las técnicas de visualización en ocasiones se utilizan para aumentar o resumir sus informes tradicionales, pero la realidad es que usando estos gráficos y técnicas de visualización puede conducir a un mayor despliegue, resultar en un entendimiento más rápido del negocio, y permite comunicar más fácilmente esta comprensión del negocio a otras personas. Estos gráficos para la visualización del conocimiento son usados en dependencia de la naturaleza de la estructura de datos del negocio a visualizar y de su estructura subyacente.

La UCI esta inmersa en el proceso de automatización de la sociedad cubana y en el desarrollo de estos proyectos se maneja un volumen de información que hacen prohibitivo su análisis de manera tradicional, producto de su complejidad, disponibilidad y las capacidades físicas limitadas de los seres humanos. Además en varios de los proyectos de exportación se prevé la confección de herramientas de soporte a las decisiones gerenciales para lo cual es indispensable contar con un método para llevar a los gerentes la información transformada en conocimiento que puedan usar en su trabajo. No existe en la UCI una herramienta que permita realizar de manera eficaz este tipo de tarea.

Esta investigación surge como necesidad de dar solución a las situaciones antes expuestas; por lo que **el problema de investigación** a resolver es *que los usuarios no son capaces de entender la información de los repositorios de datos, y los análisis inteligentes aplicados a los mismos, debido a que no existe una manera natural para la visualización del conocimiento. E*s por esto se traza como **objetivo** *proponer un mecanismo de visualización de información capaz de presentar los resultados del análisis inteligente de datos.*

Como **objeto de estudio** se han definido los mecanismos de visualización del conocimiento.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN DEL CAPITULO

Como aspecto fundamental de la Visualización de Información y a tratar en este trabajo aparecen las técnicas de representación del conocimiento, cuyo objetivo es trasformar una representación inicial de una estructura de datos en un gráfico para que esta estructura pueda ser visualmente examinada e interactuar recíprocamente con ella. Las técnicas de visualización de información son ampliamente utilizadas para mejorar nuestro entendimiento de las características de la información que se estudia. Estas técnicas se usan para el análisis, comprensión y explicación de grandes colecciones de datos

El diseño espacial y los algoritmos de representación gráfica juegan un papel fundamental en la visualización del conocimiento, un buen diseño conduce a la presentación eficiente de complejas estructuras o sistemas, mientras que un diseño pobre puede disimular la naturaleza de una estructura subyacente.

Con este fin se han definido diferentes técnicas de visualización del conocimiento que permiten que los usuarios puedan identificar fácilmente y de manera directa los patrones más significativos.

1.2 VISUALIZACIÓN COMPUTACIONAL

Podemos definir entonces la visualización de la información como la utilización de una computadora como soporte, interacción y representación visual de datos abstractos para ampliar el conocimiento. Ante grandes volúmenes abstractos de información la meta es lograr la cristalización del conocimiento, es decir, permitir a los usuarios obtener la información que necesitan y hacer que ésta tenga sentido para que puedan lograrse las decisiones en un tiempo relativamente corto. (MERCEDES VITTURINI)

Muchas clases de información no tienen una representación física obvia y natural. La visualización de información permite visualizar espacios de información abstracta, tales como

datos financieros, información de negocios, colecciones de documentos y concepciones abstractas que pueden también beneficiarse al ser presentadas en forma visual.

Como ejemplos de los objetivos de la visualización de información se pueden enunciar: mostrar tendencias en los datos; detectar discontinuidades en los mismos; identificar fácilmente máximos y mínimos; establecer límites; identificar agrupamiento en los datos; encontrar estructuras en información heterogénea y ver mucha información en una única pantalla pero al mismo tiempo ver un ítem de interés en este contexto, etc.

La visualización de información apoya el proceso de producir modelos que puedan ser detectados y abstraídos; puede reducir la búsqueda de datos al agruparlos convenientemente o al relacionar la información visualmente, permite compactar información en un espacio reducido, permite búsquedas jerárquicas mediante la utilización de vistas generales para ubicar áreas de mayor demanda. La visualización permite la recuperación de modelos de datos y estos modelos sugieren esquemas a un nivel superior. La agregación de datos se revela a través de clustering o propiedades visuales comunes.

La visualización emplea el aparato sensitivo primario humano, que es la visión, tanto como todo el poder de procesamiento de la mente humana. El resultado debe ser un medio simple y efectivo para comunicar información voluminosa y compleja de manera entendible.

1.3 VISUALIZACIÓN DE ESTRUCTURAS DE DATOS

La mayoría de las estructuras de datos de negocios están almacenadas como una simple tabla de información compuesta por un número finito de columnas y una o más filas de datos.

CLIMA

CIUDAD	FECHA	TEMPERATURA	HUMEDAD	ESTADO
Habana	01-MAY-2007	97.1	82.2	Soleado
Camaguey	01-MAY-2001	66.5	100.0	lluvioso

Santiago	01-MAY-2001	71.3	62.3	Nublado

Tabla 1.1: Ejemplo de una estructura de datos de negocio con información sobre el clima

- CLIMA es el archivo, tabla o nombre de la estructura de datos. El clima de una ciudad en un día particular es el sujeto de la investigación.
- CIUDAD, FECHA, TEMPERATURA, HUMEDAD y ESTADO son las columnas de la estructura de datos que describen la manera como se mantiene la información en la estructura de datos, describen los atributos del clima por cada ciudad.
- Habana, 01-MAY-2007, 28.2, 82.2, Nublado son un registro particular o fila en la estructura de datos.

Las herramientas y técnicas de visualización son usadas para representar gráficamente los datos de una estructura de datos como una imagen en 2D ó 3D.

La **Dimensión visual** de los datos se refiere a los ejes x-, y-, y z- del sistema de coordenadas espacial, o al color, la opacidad, la altura o el tamaño del objeto gráfico. Por otro lado la **Dimensión de los Datos** se refiere al número de columnas o las variables discretas o continuas, contenidas en la estructura de datos de negocio.(DAVIDSON May 2002)
Si utilizamos la estructura de datos de negocio de la tabla 1.1, la **Dimensión de los datos** de la estructura de datos del CLIMA serian las columnas CIUDAD, FECHA, TEMPERATURA, HUMEDAD Y ESTADO. Para crear una visualización de 2D o 3D de la estructura de datos del CLIMA, se seleccionan las columnas de la estructura para crear una tabla gráfica de datos. La tabla gráfica de datos es usada para mapear los valores de las columnas de la estructura con sus puntos correspondientes en un sistema de ejes coordenados x-, y-, ó z-.
La Fig. 1.1 ilustra una visualización de un gráfico de columnas comparando la TEMPERATURA y la HUMEDAD continuos por CIUDAD para la estructura de datos del CLIMA. La tabla gráfica de valores correspondiente con las columnas de TEMPERATURA y HUMEDAD está representada

por la altura de las barras de la gráfica de columnas. Un par de barras se dibujan para cada valor de la CIUDAD.

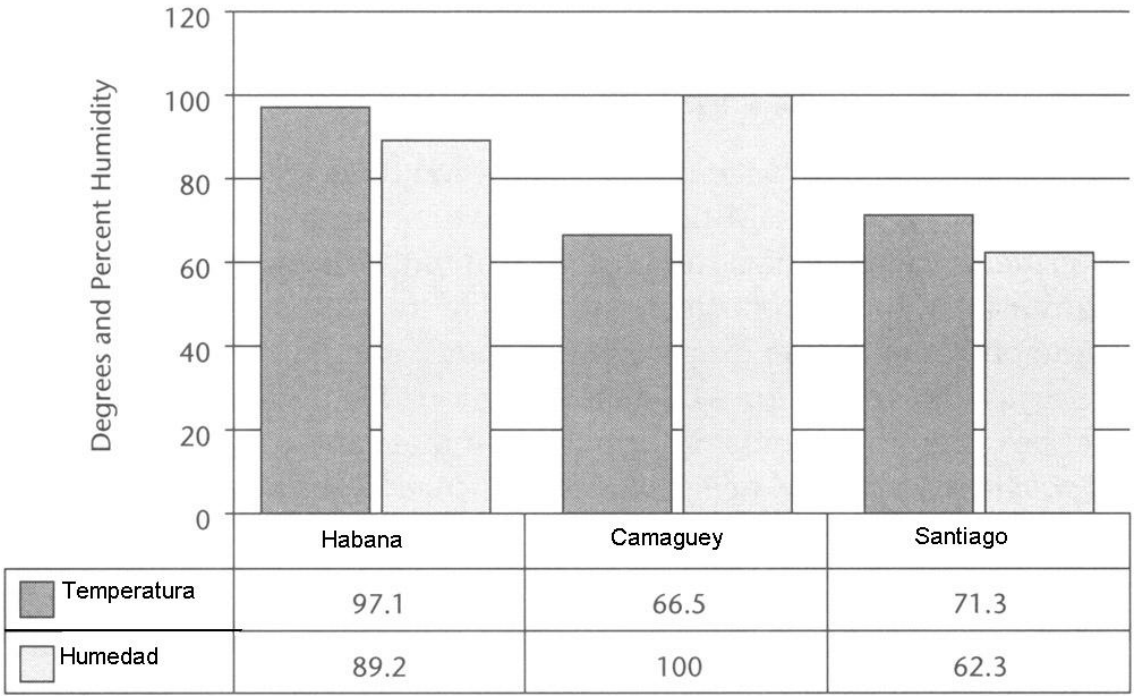


Fig. 1.1 Gráfico de columnas comparando temperatura y humedad por ciudad.

1.4 GRÁFICOS DE VISUALIZACIÓN DE DATOS

Los gráficos de visualización de datos son usados para crear visualizaciones de 2 ó 3 dimensiones de una estructura de datos de negocio. Algunos de estos gráficos permiten incluso animar la visualización a través de una o más dimensiones de datos. Los más simples gráficos para visualizar la información, tales como, líneas, barras, columnas y gráficos de pastel han sido usados por siglos. Algunas empresas usan estas técnicas de visualización para aumentar o resumir sus informes tradicionales, pero la realidad es que usando estos gráficos y técnicas de visualización puede conducir a un mayor despliegue, resultar en un entendimiento más rápido del negocio, y permite comunicar más fácilmente esta comprensión del negocio a otras personas. Estos gráficos para la visualización del conocimiento son usados en dependencia de la

naturaleza de la estructura de datos del negocio a visualizar y de su estructura subyacente. Los gráficos para la visualización del conocimiento pueden ser clasificados en dos categorías principales:

- Visualización multidimencional.
- Visualización especializada de Jerarquías.

1.4.1 VISUALIZACIÓN MULTIDIMENCIONAL

Los gráficos de visualización más comúnmente usados son aquellos que representan gráficamente estructuras de datos multidimencionales. Los gráficos de visualización multidimencional de los datos permiten a los usuarios comparar visualmente las dimensiones de los datos, valores de las columnas con otras dimensiones de datos usando sistemas de coordenadas espaciales. (DAVIDSON May 2002)

En la **Fig. 1.2** se muestran ejemplos de los tipos de gráficos de visualización más comúnmente usados.

Fig. 1.2 Gráficos de visualización multidimensional.

Muchos gráficos de visualización multidimensional de datos son usados para comparar y contrastar los valores una columna o dimensión de datos con los valores de otra columna en la estructura de datos. Estos son también usados para investigar la relación entre dos o más columnas continuas o discretas en la estructura de datos. En la tabla 1.2 se listan algunos tipos de gráficos multidimensionales y los tipos de valores de dimensión de datos que estos pueden comparar, o los tipos de relaciones que pueden investigar.

GRÁFICOS	**VALORES DE COLUMNAS A COMPARAR**
Barras , Columnas	Usado para comparar los valores discretos

	(categóricos) de la columna con los valores continuos de la columna (dimensión de datos), por ejemplo, compara los datos entre diferentes segmentos (sectores, empresas, períodos de tiempo).
Área, Radar, Líneas	Usados para comparar valores discretos (categóricos) de una columna sobre una columna continua, por ejemplo, para comparar entidades (dos productos que presentan varias características pueden ser comparados usando estas gráficas).
Pastel, Rosquilla, Histograma, Distribución	Usadas para comparar la distribución de valores distintos para una o más columnas discretas.
Dispersión	Usadas para investigar la relación entre dos o más columnas continuas.

Tabla 1.2

1.4.1.1 Gráfico de Columnas y barras

Los gráficos de columnas y barras comparan dimensiones de datos continuas a través de dimensiones de datos discretas en un sistema de coordenadas x-, y-. Los gráficos de columnas, al igual que los gráficos de líneas representan gráficamente muchas dimensiones de datos, con la excepción de que las columnas verticales se representan desde el eje x-, hacia el eje y- del sistema de coordenadas hasta el valor de la dimensión de datos. Los gráficos de barras son idénticos a los gráficos de columnas, con la excepción de que los ejes x- y y- están intercambiados de manera que las entidades gráficas son representadas horizontalmente en lugar de verticalmente. En cualquier caso los valores de datos asociados con diferentes estructuras de datos son agrupados por su posición en el eje x- para facilitar la comparación

entre los grupos. De esta manera las entidades gráficas aparecen agrupadas una al lado de otra. Cada estructura de datos puede ser representada por un color o patrón diferente para su mejor entendimiento.

La **Fig. 1.1** ilustra una visualización multidimensional de un gráfico de columnas comparando la dimensión de datos TEMPERATURA y HUMEDAD con la dimensión de datos CIUDAD para la estructura de datos CLIMA desde la **Tabla 1.1.**

1.4.1.2 Histograma y Gráfico de Distribución

Una técnica analítica extremadamente útil es usar los gráficos básicos de barras y columnas para representar la distribución de los valores para una dimensión de datos (columna). Los gráficos de distribución e histogramas representan la proporción de los valores para columnas discretas (no-numéricas) y continuas (numéricas) como los gráficos especializados de barras y columnas. Un gráfico de distribución muestra la ocurrencia de los valores discretos (no-numéricos) de la columna en una estructura de datos. Un uso típico de los gráficos de distribución es demostrar desequilibrios en los datos. Un histograma también llamado como gráfico de frecuencia representa en número de ocurrencias de un mismo o distintos valores en la estructura de datos. Estos también son usados para revelar desequilibrios en los datos.

La **Fig. 1.3** ilustra un gráfico de distribución de la dimensión de datos de la FECHA de FACTURA de registros para los primeros 4 meses del 2007. El gráfico de distribución te proporciona un método para verificar si hay registros perdidos en la estructura de datos de negocio.

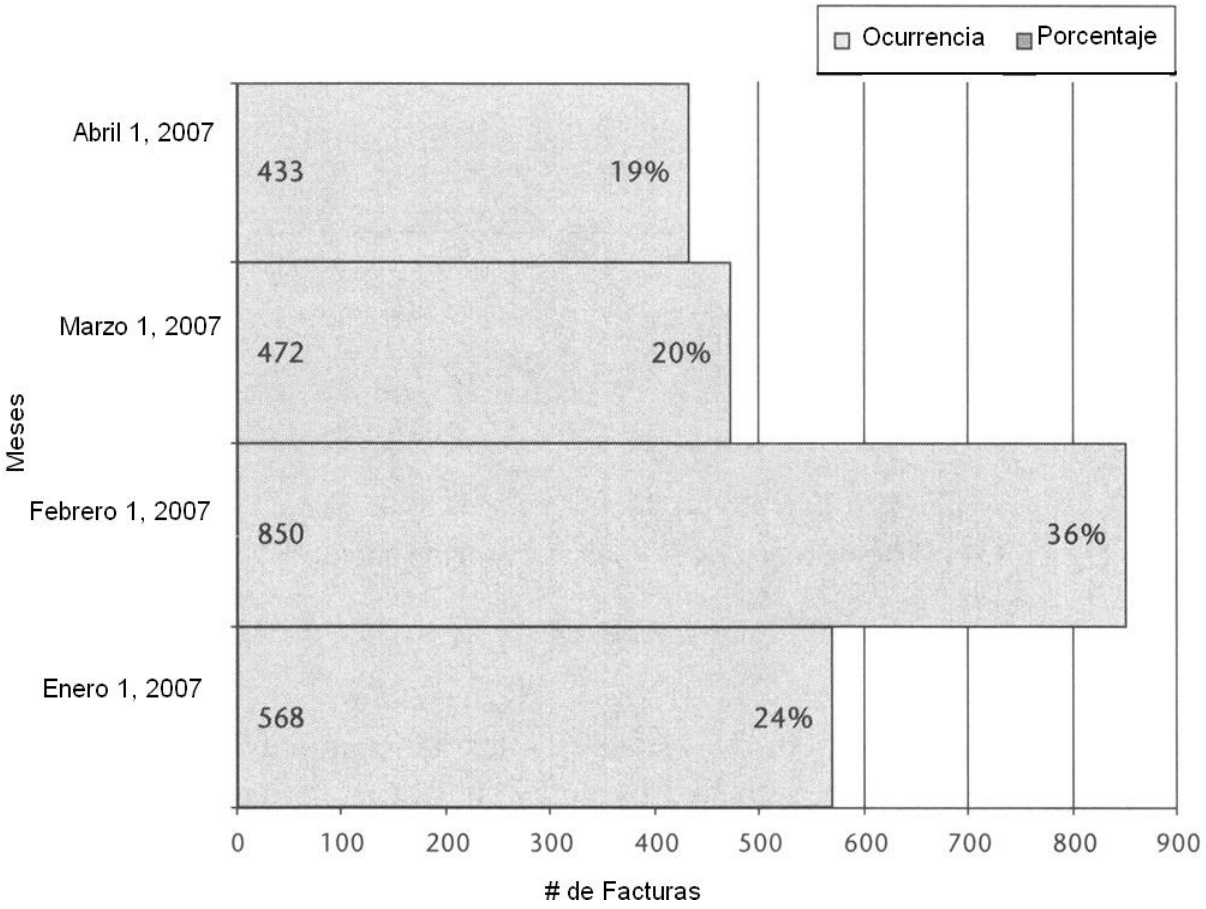


Fig. 1.3 Gráfico de Distribución de facturas para los primeros 4 meses del 2007.

La **Fig. 1.4a** muestra un gráfico de Histograma para el número de facturas por REGION y en la **Fig. 1.4b** se muestra un gráfico de Histograma para el número de facturas por la TARIFA DE FACTURACION agrupados para los primeros 4 meses del 2007 desde la misma estructura de datos de contabilidad. En ambos gráficos podemos ver visualmente la oblicuidad (carencia de simetría en una distribución de frecuencia) en la distribución de los valores de la columna. Por ejemplo, el gráfico de Histograma de la **Fig. 1.4a** se inclina hacia la región del ESTE mientras que en el gráfico de la **Fig. 1.4b** se inclina hacia la tarifa de facturación de $15.00 por hora o menos.

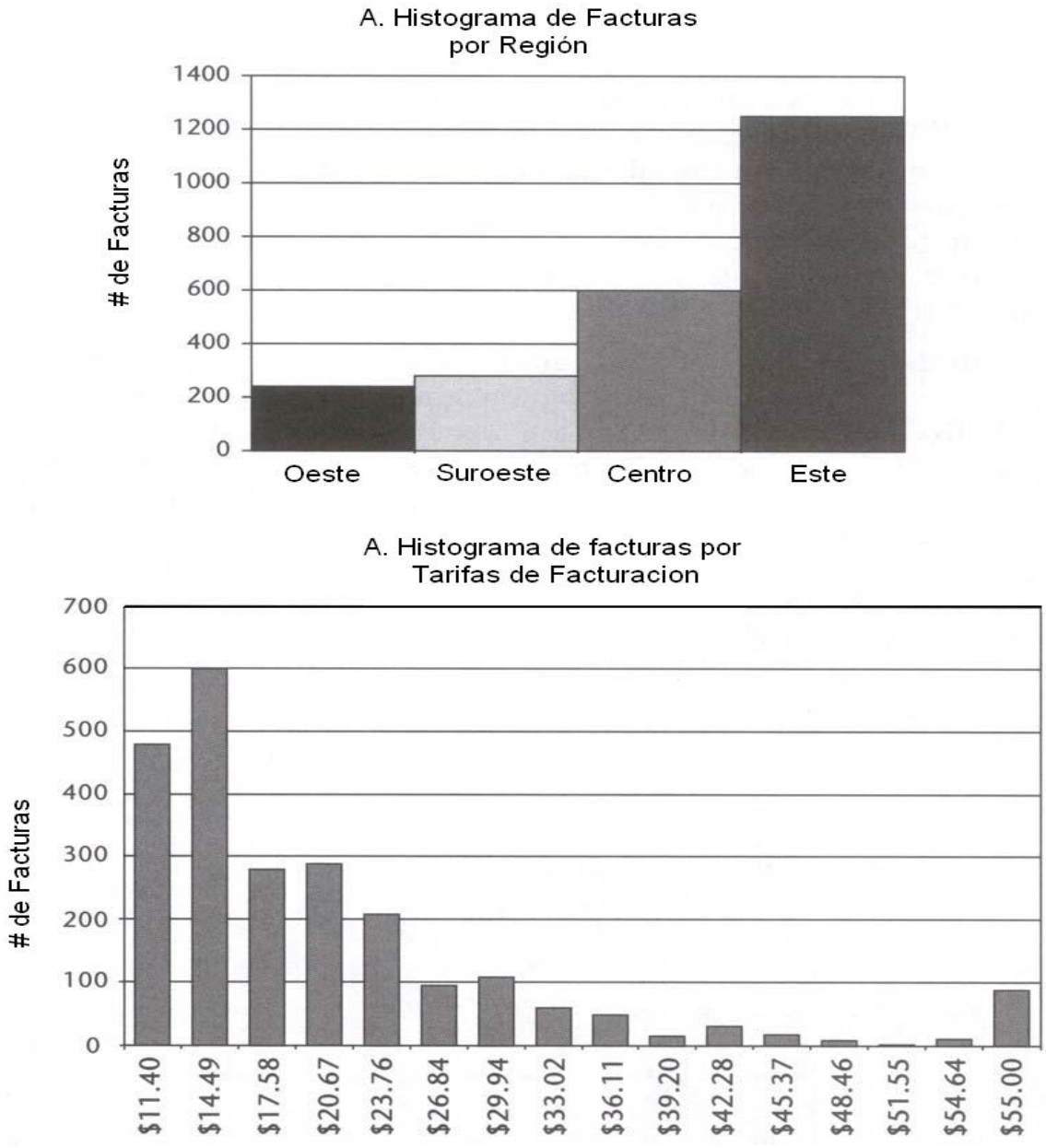


Fig. 1.4 Gráficos de Histograma de facturas por región y tarifas de facturación.

1.4.1.3 GRÁFICO DE LÍNEA

En su forma más simple un gráfico de líneas no es más que un sistema de puntos de referencia representados gráficamente en un sistema de coordenadas x-, y-, conectados posiblemente por

segmentos de línea. Los gráficos de línea normalmente muestran los valores de una columna comparados con otra columna (dimensión de datos) dentro de un sistema de coordenadas x-, y-. Los segmentos de línea conectarán puntos adyacentes desde los valores de la columna de datos. Los valores de los datos para el eje x-, pueden ser discretos o continuos, si son discretos, los valores se convierten en las etiquetas para las localizaciones sucesivas en el eje. Los valores de los datos para el eje y-, deben ser continuos. A menudo los gráficos de línea son usados para demostrar tendencias en series de tiempo.

En la Fig. 1.5 se muestra la visualización de un gráfico de línea comparando los índices de producción en enlace de 1-, 2-, 3-, y 12 meses desde 1/17/2002 hasta 6/23/2006. La dimensión de los datos de la serie de tiempo (FECHA) esta representada en el eje x-. Los valores de los datos correspondientes para las producciones de 1-, 2-, 3-, y 12 meses están representados en el eje y-. Los valores correspondientes de las columnas de datos están mostrados como puntos conectados por una línea dentro del sistema de coordenadas x-, y-.

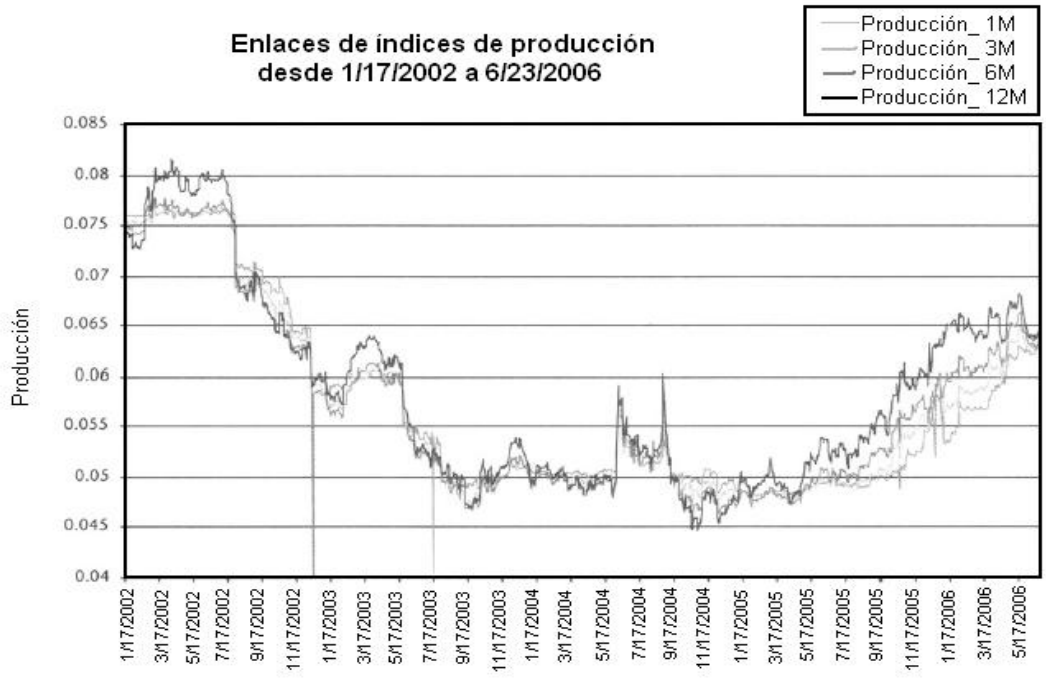


Fig. 1.5 Gráfico de línea de los enlaces de índices de producción.

1.4.1.4 GRÁFICO DE RADAR

El gráfico de radar es una variación del gráfico de línea, este muestra los radares con los marcadores en cada punto de referencia en sistema de coordenadas de 360-grados en lugar del sistema coordenado x-, y-, tradicional de 90-grados.

La **Fig. 1.6** muestra un gráfico de radar comparando los índices de producción en enlaces de 1- y 6- meses.

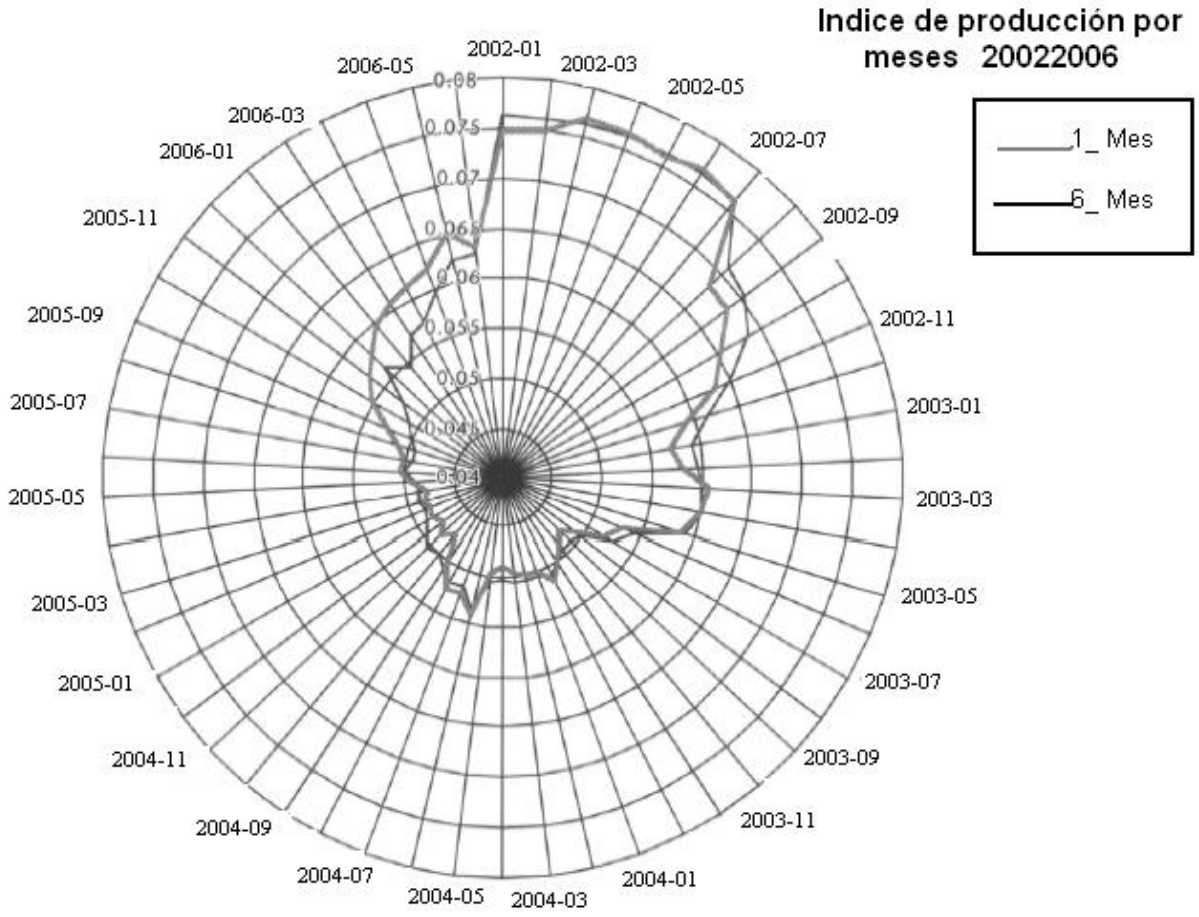


Fig. 1.6 Gráfico de Radar de los enlaces de índices de producción.

1.4.1.5 GRÁFICO DE DISPERSIÓN

Los gráficos de dispersión (a menudo llamados como diagramas de dispersión) son típicamente usados para comparar pares de valores. Un gráfico de dispersión permite visualizar una estructura de datos de negocio mapeando cada fila o registro de la estructura de datos para una entidad gráfica dentro de un gráfico de dos o tres dimensiones. En contraste con los gráficos de línea, un gráfico de dispersión muestra puntos de referencia no relacionados en un sistema de coordenadas x-, y-, o z. El gráfico de burbujas es una variación de un gráfico simple de dispersión que permite representar otra dimensión de datos de la estructura como el tamaño de la entidad gráfica, así como también su posición dentro del sistema de coordenadas x-, y-.

La **Fig. 1.7** ilustra un ejemplo de como se puede usar un gráfico de dispersión para investigar la relación entre el número de promociones de un almacén y su beneficio semanal.

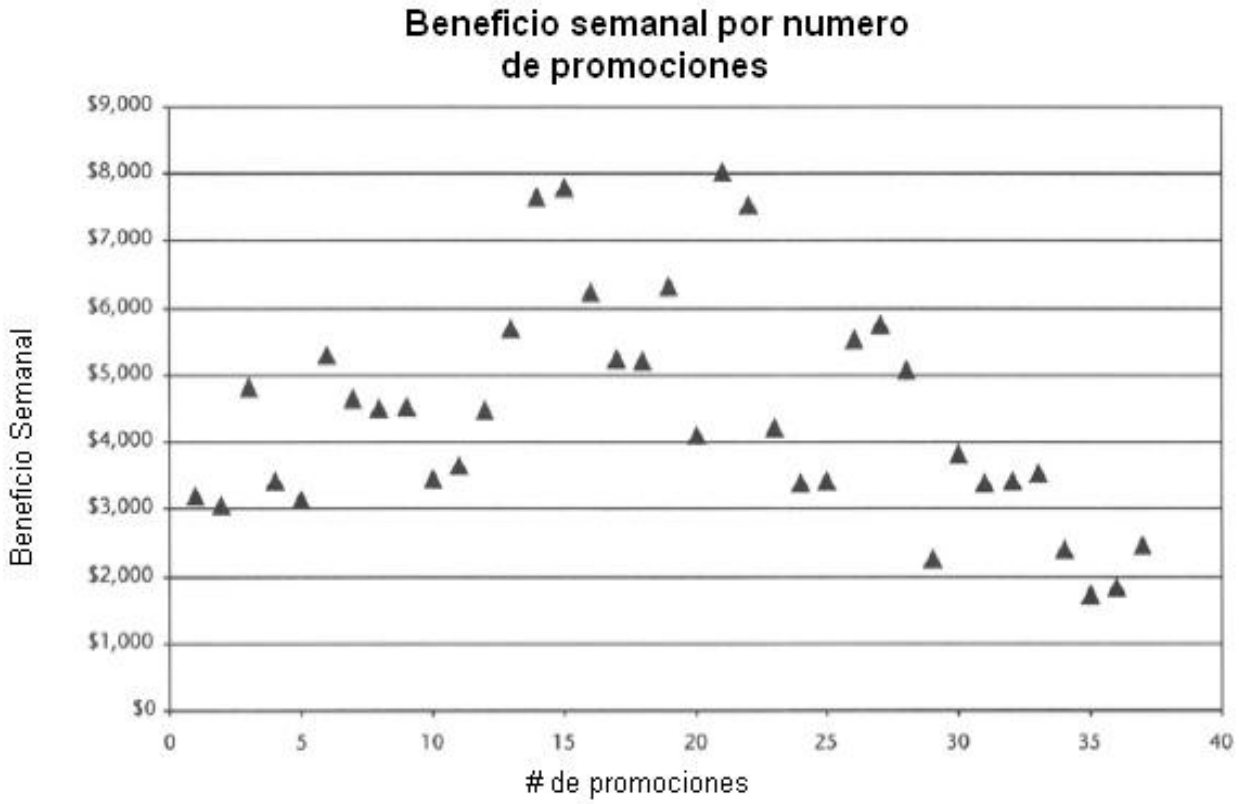


Fig. 1.7 Gráfico de Dispersión para el beneficio semanal de un almacén por número de promociones.

1.4.1.6 GRÁFICO DE PASTEL

El gráfico de pastel muestra la contribución de cada valor con respecto a la suma de los valores para una columna (dimensión de datos) en particular. Los valores discretos de la columna se convierten en las etiquetas para las rebanadas del gráfico de pastel, mientras que los valores continuos de la columna se resumen dentro de la contribución a través de los valores discretos de la columna.

La **fig. 1.8a** muestra un gráfico de pastel comparando la contribución del porcentaje de la población de estudiantes de ambos sexos y profesores en la UCI.

El gráfico de rosquilla es una variación del gráfico de pastel, este puede ser usado para comparar múltiples columnas de valores continuas al mismo tiempo. Por ejemplo en la **fig1.8b** se muestra una comparación de la distribución del porcentaje de población de estudiantes de ambos sexos y profesores en la UCI, pero este gráfico no solo permite hacer esta comparación de forma simple, sino que usándolo se podría hacer esta comparación para la UH, IPSJAE y la UCI dentro de la misma visualización.

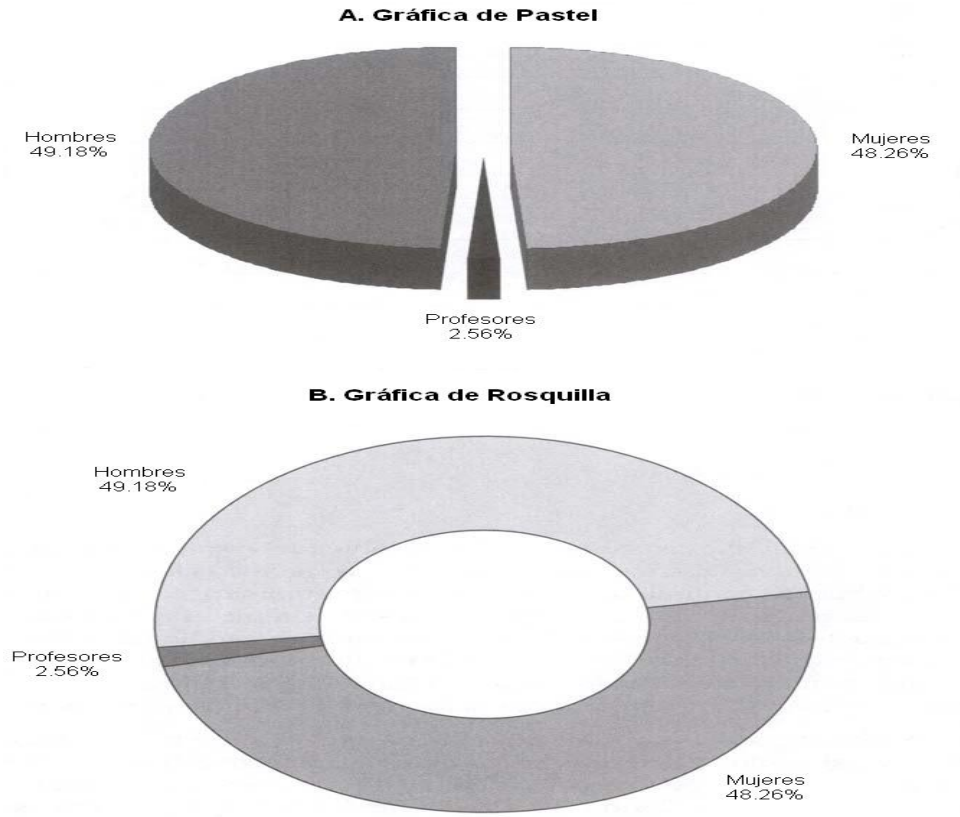


Fig. 1.8 Gráfico de Rosquilla y de Pastel para el porcentaje de población de estudiantes de ambos sexos y profesores en la UCI.

1.4.2 VISUALIZACIÓN ESPECIALIZADA DE JERARQUÍAS

Las técnicas de visualización especializada de jerarquías difieren de las técnicas comunes de visualización multidimencional en que estas explotan o resaltan la estructura subrayada de su propio conjunto de datos del negocio.(DAVIDSON May 2002)

Algunas estructuras de datos de negocio poseen una estructura jerárquica inherente que puede ser visualizada mucho más fácilmente utilizando las técnicas de visualización especializada de jerarquías. La visualización de árboles de decisión y mapas conceptuales pueden ser usados para explorar las relaciones entre los niveles jerárquicos de un conjunto de datos.

1.4.2.1 ÁRBOLES DE DECISIÓN

Las decisiones que tomemos hoy, por lo general, condicionan nuestro futuro de cierta manera. Esto hace que las decisiones no estén normalmente aisladas y que cada una de ellas desencadene una serie de consecuencias en muchos casos inciertas. Si elegimos una alternativa habrá en consecuencia un curso de acción posible, y si no elegimos dicha alternativa tendremos a nuestro alcance quizás otro curso de acción. Observe que tanto actuar como no actuar son aspectos básicos de múltiples procesos decisorios.

Los árboles de decisión son, seguramente, los modelos que permiten una representación visual más clara gracias a la propia estructura en árbol de los modelos. Un árbol de decisión puede verse como un grafo parcialmente ordenado donde los nodos sólo tienen un padre. Dado que un árbol puede ser de gran tamaño, muchas herramientas permiten mostrar segmentos parciales del árbol, empezando por las ramas superiores, y desplegar las partes que el usuario seleccione hasta llegar a las hojas.

Algunas estructuras de datos de negocio poseen una estructura jerárquica inherente, la visualización usando la técnica de árboles de decisión puede ser muy conveniente para explorar las relaciones entre los niveles jerárquicos. Un árbol de decisión representa la estructura de datos en forma de árbol, cada nivel en las ramas del árbol esta basado sobre los valores de diferentes atributos del problema con una jerarquía en la estructura de datos. El gráfico de árbol presenta cuantitativamente y relacionalmente las características de la estructura de datos mostrándola como nodos conectados jerárquicamente. Cada nodo contiene una información del problema usualmente en forma de barra o disco cuyo nivel y color se corresponde con la agregación de los valores de los datos. Las líneas, usualmente llamadas bordes, conectan los nodos y muestran la relación de un conjunto de datos con sus subconjuntos.

Las ventajas de un árbol de decisión son:

- Ayudan a construir una imagen balanceada de los riesgos y recompensas asociados con cada posible curso de acción.

- Claramente plantean el problema para que todas las opciones sean analizadas.

- Permiten analizar totalmente las posibles consecuencias de tomar una decisión.

- Proveen un esquema para cuantificar el costo de un resultado y la posibilidad de que suceda.

- Ayuda a realizar las mejores decisiones sobre la base de la información existente y de las mejores suposiciones.

- Reduce el número de variables independientes.

- Es una magnifica herramienta para el control de la gestión empresarial.

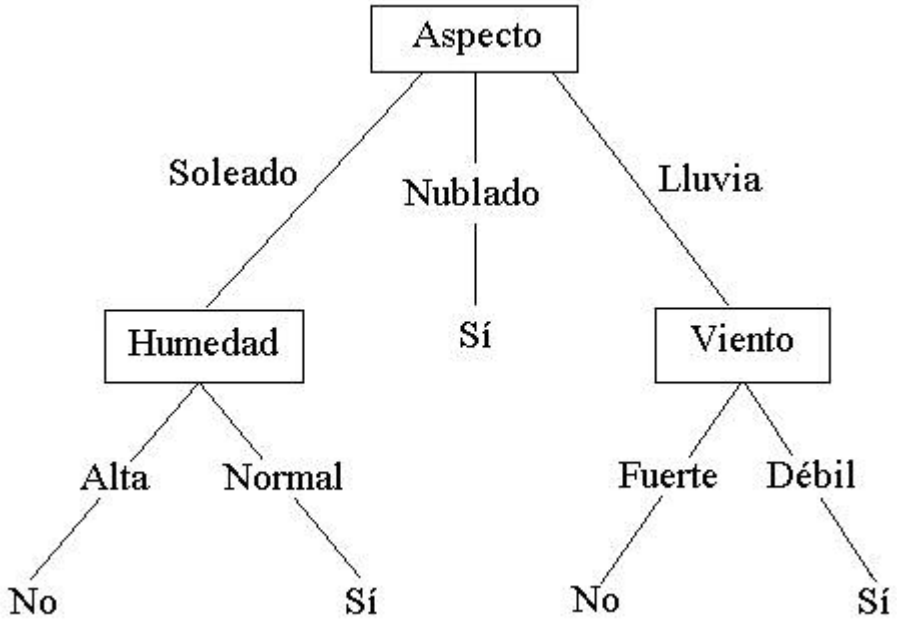


Fig. 2. Ejemplo de árbol de decisión típico para decidir "*si es un buen día*", Los nodos representan un atributo a ser verificado por el clasificador. Las ramas son los posibles valores para el atributo en cuestión.

1.4.2.2 MAPAS CONCEPTUALES

Mapa Conceptual es una técnica usada para la representación gráfica del conocimiento, un mapa conceptual es una red de conceptos. En la red, los nodos representan los conceptos, y los enlaces, las relaciones entre los conceptos.

Los mapas surgieron en el ámbito de la didáctica de las disciplinas científicas de la mano de Novak y Gowin en 1984, quienes expresaron que "los mapas conceptuales tienen por objeto representar relaciones significativas entre conceptos en forma de proposiciones. Una proposición consta de dos o más términos conceptuales unidos por palabras para formar una unidad semántica." (JOSEPH NOVAK 1984)

El diagrama muestra la relación entre los conceptos y estos están relacionados mediante flechas etiquetadas en una ramificación descendente de una estructura jerárquica. La relación de los

conceptos es articulada en conectar frases, por ejemplo: “dar origen a”, “da lugar a”, “es requerido por” o “contribuye a”.

Los mapas conceptuales son usados para estimular la generación de ideas y ayudar a la creatividad, para reuniones creativas, pueden ser utilizados también para comunicar ideas complejas, facilitar la creación de la visión compartida y de la comprensión compartida dentro de un equipo o de una organización, para representar el contexto del entrenamiento y su relación a sus trabajos, representar los objetivos estratégicos de la organización y sus metas, para mejorar la comprensión de los objetivos, conceptos dentro de la organización y la relación entre estos conceptos.

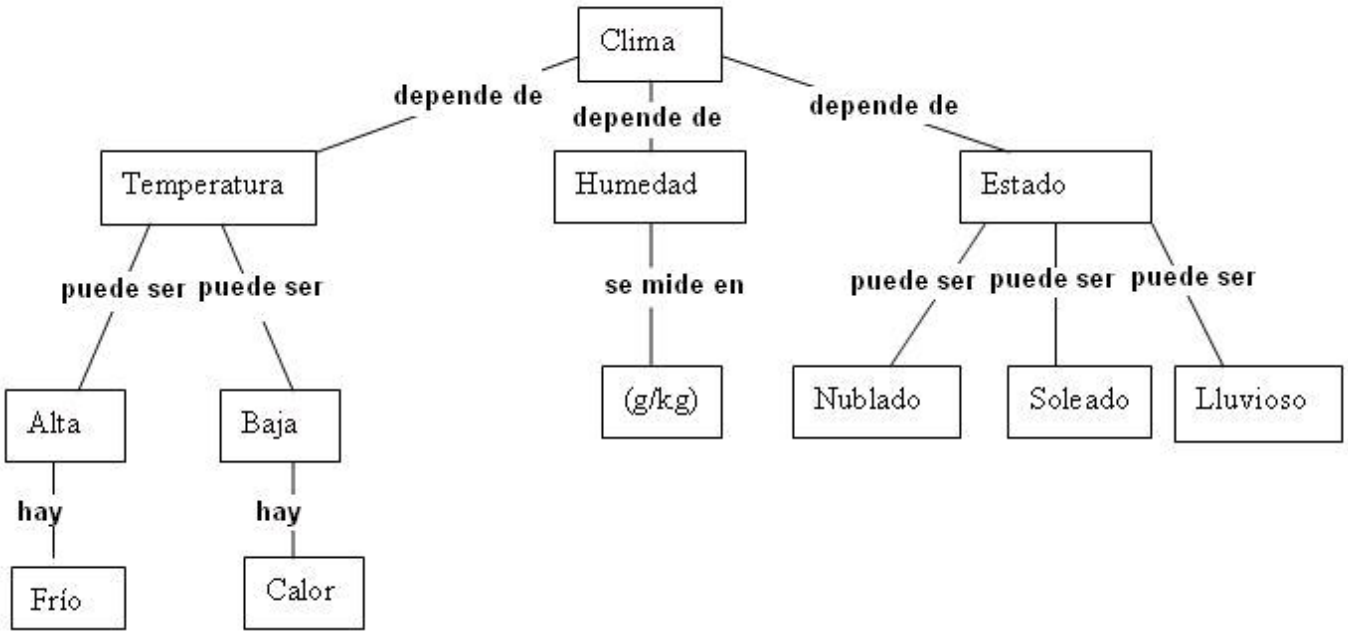


Fig. 2. Ejemplo de un mapa conceptual.

1.5 ESTÁNDARES DE REPRESENTACIÓN

Existe un interés creciente en la definición y uso de estándares para intercambiar información entre aplicaciones de distintos tipos. Un estándar es el conjunto de reglas y especificaciones a seguir, desarrolladas, de común acuerdo, para su uso por cualquier empresa, institución o persona que representa cualquier sector industrial y tiene como fin cubrir una necesidad vigente.(AMECE 2007)

La definición del lenguaje XML (Extensible Markup Language) ha impulsado su utilización para la creación de lenguajes estándar para intercambio de información. XML es un lenguaje de marcas definido por el consorcio World Wide Web Consortium (W3C, http://www.w3.org/).

XML puede considerase más que un lenguaje como un meta-lenguaje ya que permite la definición de lenguajes de marcas para diferentes tipos de documentos.(JOSÉ ANTONIO LÓPEZ PÉREZ abril 2002)

XML no ha nacido solo para su aplicación en Internet, sino que se propone como lenguaje de bajo nivel, a nivel de aplicación y no de programación, para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, y casi cualquier cosa en la que se pueda pensar. Estos lenguajes definidos en XML, recorren áreas como la química y la física, las matemáticas, el dibujo, entre muchas otras. XML contiene datos e información y permite definir etiquetas (tags) propias que permiten personalizar un lenguaje de marcas a los diferentes tipos de documentos.

Con ese fin XML utiliza las DTD (Document Type Definition) para describir las características de cada tipo de documentos. Las DTD son una definición de los elementos que puede haber en el documento XML, y su relación entre ellos, sus atributos, posibles valores, etc. En definitiva, es una especie de definición de la gramática del documento. Cuando se procesa cualquier información formateada mediante XML, lo primero es comprobar si esta bien formada, o sea, que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas propias de XML, y luego si incluye o referencia un DTD.

Existen quizás varias iniciativas para establecer estándares de intercambio de información, pero sin duda alguna la iniciativa más destacada es PMML (Predictive Model Markup Language), este

estándar de la industria fue desarrollado por el Data Mining Group (http://www.dmg.org/) establecido en 1998, un grupo independiente formado por más de 25 compañías líderes en tecnología, como por ejemplo: IBM, Microsoft, Oracle, SAP, SAS, SPSS...entre otras. PMML se basa en XML. El lenguaje soporta, entre otros, los siguientes tipos de modelos: regresión polinomial y general, árboles de decisión, agrupamiento basado en centros y en distribuciones, reglas de asociación y secuencias, etc, y cada tipo de modelo se expresa mediante un documento DTD o un XML Schema.

Utilizando este estándar los desarrolladores de aplicaciones encontrarán más fácil interactuar entre diversos sistemas, y así los analistas pueden definir modelos de predicción y compartir estos modelos con otras aplicaciones que lo soportan, que por su parte serán capaces de implantar más fácilmente estos modelos en herramientas que den soporte, tales como los Sistemas de Visualización de Información.(JOSÉ ANTONIO LÓPEZ PÉREZ abril 2002)

1.6 Marco Metodológico. Métodos y técnicas utilizadas.

Esta investigación contiene un estudio de las técnicas de visualización de la información y muestra una vía para trasformar y visualizar de forma gráfica el resultado de un algoritmo aplicado a un conjunto de datos.

La estrategia de investigación que se utilizó de acuerdo al objetivo de este trabajo es la investigación exploratoria Este tipo de investigación se realiza cuando existe una problemática que esta afectando la sociedad y no se tiene una idea clara del asunto en cuestión.

Dentro de los métodos a utilizar están los de carácter teórico que incluye el método Analítico sintético y el método empírico dentro de este la observación científica. El diseño de investigación intenta dar respuesta a la problemática de mostrar la información de forma gráfica de modo que esto conlleve a un mejor entendimiento de los datos almacenado en los repositorios.

1.7 CONCLUSIONES DEL CAPITULO

Un mecanismo de visualización debe ser capaz de representar la información en forma gráfica teniendo en cuenta las categorías o clasificaciones y técnicas para la visualización del conocimiento expuestas anteriormente en este capítulo. El mundo tiende hacia la utilización de estándares abiertos de Internet en la construcción de los sistemas, en el campo de la visualización el estándar imperante en el mercado es el PMML y por tanto seria recomendable la utilización del mismo en el diseño.

Capítulo 2 Estándares de Representación

2.1 Introducción del capitulo

En este capitulo se pretende hacer un estudio acerca de PMML, estándar imperante en el campo de la visualización de la información, el cual es utilizado como información de entrada de esta mecanismo para visualizar la información producto del análisis inteligente de los datos. Se describe la arquitectura de un documento PMML, así como sus principales elementos. Se incluye además un estudio acerca del estándar SVG utilizado para describir gráficas.

2.2 Visión general de PMML

Predictive Model Mark-up Language (PMML) es un estándar basado en XML, el cual suministra un camino para definir modelos estadísticos y de data mining y para compartir estos modelos entre aplicaciones con soporte PMML.(GROUP)

PMML provee métodos para que los problemas entre el software propietario y la incompatibilidad entre aplicaciones no sigan siendo una barrera para intercambiar modelos. PMML, formalmente definido como XML Schema, permite a los usuarios desarrollar modelos utilizando una aplicación determinada y posteriormente usar otra para analizar, visualizar y evaluar estos modelos. Usando PMML el intercambio de modelos entre aplicaciones es mucho más viable, en el epígrafe 2.3 abordamos las ventajas que trae consigo la utilización de este estándar, anteriormente esto era muy difícil ya que no existía un modo de intercambio entre diferentes aplicaciones o sistemas para estos modelos.

Más específicamente, PMML tiene como objetivo definir y compartir un modelo predictivo[1] de manera que dos aplicaciones diferentes (el productor PMML y el consumidor) puedan usarlo sin problema.

[1] Modelo predictivo: es un proceso usado para crear modelos lógicos de un comportamiento futuro

2.3 Ventajas de PMML

- Este suministra independencias del conocimiento extraído de la aplicación, implementación, plataforma y sistema operativo.
- Facilita el uso de los modelos de minería de datos para otras aplicaciones o personas.
- No es afectado con el proceso de creación del modelo o con la implementación especifica del algoritmo.
- Las DTDs dan soporte a extensiones propietarias lo que permite enriquecer la información almacenada para herramientas especializadas. La DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD se denominan válidos.

2.4 Estructura de un documento PMML

PMML usa XML para representar modelos de minería de datos. La estructura del modelo es descrita por un XML Schema (Document type Definition). Uno o más modelos PMML pueden estar contenidos dentro de un documento PMML, este es un documento XML con un *root element* de tipo PMML. La estructura general de un documento PMML es:

- Estructura Básica de XML
- Root Element <PMML>
    - Namespaces
- Elementos hijos
        - Header
        - Data Dictionary
        - MiningBuildTask
        - TransformationDictionary
        - Secuencia del modelo

- Extensión

```
<?xml version="1.0"?>
    <PMML version="3.1"
xmlns="http://www.dmg.org/PMML-3_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
        <Header … />
        <DataDictionary>... </DataDictionary>
        ….. el modelo…..
    </PMML>
```

2.4.1 NAMESPACE

Un *Namespace* es una colección de nombres identificada por una referencia URL. Su propósito es desarrollar un vocabulario XML (en el cual son definidos elementos y atributos) en un ambiente global y reducir el riesgo de colisiones de nombres en determinado documento cuando los vocabularios son combinados. El atributo del *namespace* de XML (xmlns) se pone en la etiqueta de comienzo de un elemento y posee la sintaxis siguiente:
xmlns: prefix=”namespaceURL”.

Ejemplo:
xmlns=”http://www.dmg.org/PMML-3_1”
xmlns: xs=”http://www.w3.org/2001/XMLSchema”

2.4.2 ROOT ELEMENT

Un documento PMML debe ser válido con respecto al PMML XSD (XML Schema Definition), además debe obedecer a un número de reglas las cuales están descritas en la especificación del documento PMML.

El *root element* de un documento PMML debe tener el tipo PMML:

```
<xs:element name="PMML">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Header"/>
   <xs:element ref="MiningBuildTask" minOccurs="0"/>
   <xs:element ref="DataDictionary"/>
   <xs:element ref="TransformationDictionary" minOccurs="0"/>
   <xs:sequence minOccurs="0" maxOccurs="ilimitado">
    <xs:choice>
     <xs:element ref="AssociationModel"/>
     <xs:element ref="ClusteringModel"/>
     <xs:element ref="GeneralRegressionModel"/>
     <xs:element ref="MiningModel"/>
     <xs:element ref="NaiveBayesModel"/>
     <xs:element ref="NeuralNetwork"/>
     <xs:element ref="RegressionModel"/>
     <xs:element ref="RuleSetModel"/>
     <xs:element ref="SequenceModel"/>
     <xs:element ref="SupportVectorMachineModel"/>
     <xs:element ref="TextModel"/>
     <xs:element ref="TreeModel"/>
    </xs:choice>
   </xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="ilimitado"/>
  </xs:sequence>
  <xs:attribute name="version" type="xs:string" use="requerido"/>
 </xs:complexType>
</xs:element>
```

Un documento PMML puede contener más de un modelo. Si la aplicación que genera el documento PMML provee una manera de seleccionar los modelos por nombre y el consumidor PMML especifica el nombre del modelo, entonces este es usado, de otra manera es usado el primer modelo. La lista de modelos en el documento PMML puede incluso estar vacía, pero este no sería de utilidad para el consumidor PMML.

2.5 PMML Composición:

2.5.1 Header

Este elemento define una descripción general del documento PMML, en el cual se definen los siguientes atributos:

***Copyright*:** Este atributo contiene la información del *copyright* para el modelo.

Description: Este atributo contiene una descripción no específica acerca del modelo. Contiene información necesaria para el uso del modelo en futuras aplicaciones.

Application: Este elemento se describe el nombre y la versión de la aplicación que generó el modelo. Aunque un modelo PMML es creado para ser portable, diferentes mecanismos pueden crear diferentes modelos desde la misma estructura de datos. Por esta razón es de interés para el usuario saber desde qué aplicación fue generado el modelo.

Annotation: Cada anotación es un texto libre, donde se introduce una pequeña descripción acerca de la creación del modelo PMML.

Timestamp: fecha y hora de creación del modelo.

Estructura del elemento Header:

```
<Header copyright="  " description="  " >
   <Application name="nombre" version=" " >
   <Annotation>….texto libre …..</Annotation>
   <Timestamp>…..fecha/tiempo de creación del
                 modelo……</Timestamp>
</Header>
```

2.5.2 MiningBuildTask

El elemento *MiningBuildTask* puede contener cualquier valor XML describiendo la configuración de la ejecución que produjo la instancia del modelo. La información que se suministra en este elemento es esencialmente *metadata*. Esta información no es usada específicamente en el desarrollo del modelo por el consumidor PMML, pero en muchos casos es de mucha ayuda para el mantenimiento y visualización del modelo. El contenido particular de la estructura de *MiningBuildTask no es definido por PMML.*

Estructura del elemento MiningBuildTask

```
<xs:element name="MiningBuildTask">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
```

2.5.3 Data Dictionary

Este elemento contiene las definiciones de los campos usados en el modelo. Especifica los tipos y rangos de los valores. En este elemento se definen los siguientes atributos

NumberOfFields: Muestra el número de campos (*datafield)* que son definidos en el contenido de *DataDictionary*, debe ser un número entero no negativo.

Datafield: Su nombre (*name*) debe ser único en el *datadictionary*, y se define por un string, el cual es usado por las aplicaciones para referirse a un datafield específico, estos campos son separados en diferentes tipos (*optype*) los cuales pueden ser: categóricos, ordinales y continuos. Puede haber tantos *datafield* como sea posible.

El contenido del *DataField* define un conjunto de valores que se consideran válidos:

Valores usables: El valor de la entrada falta, por ejemplo, si una columna de la base de datos contiene un valor nulo. Es posible definir explícitamente los valores que se interpretan como valores que faltan.

Valor no válido: El valor de la entrada no pertenece a cierta gama de valores. La gama de valores válidos se puede definir para cada campo.

Valor válido: es un valor que no esta faltando ni es no válido

Estructura del elemento DataDictionary:

```
<xs:element name="DataDictionary">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="DataField" maxOccurs="unbounded"  />
   <xs:element ref="Taxonomy" minOccurs="0" maxOccurs="unbounded"  />
  </xs:sequence>
  <xs:attribute name="numberOfFields" type="xs:nonNegativeInteger" />
 </xs:complexType>
</xs:element>
```

2.5.4 TRANSFORMATIONDICTIONARY

Los modelos de minería utilizan funciones simples para transformar los datos de los usuarios a los valores que son más fáciles de utilizar por un modelo específico.

DeriveField: Los elementos correspondientes a XML aparecen en este elemento, el cual, proporciona un elemento común para varias representaciones de un modelo.

DefineFuncion: PMML proporciona un grupo de funciones predefinidas que soportan transformaciones como por ejemplo, cambiar caracteres de minúscula a mayúsculas o convertir los valores de fecha y hora en valores *strings,* estas funciones son llamadas *DefineFuncion*

Estructura del elemento TransformationDictionary

```
<xs:element name="TransformationDictionary">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="DefineFunction"  minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="DerivedField" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

PMML define varias clases de transformaciones simples de los datos:

- **Normalización:** proporcionan un marco básico para los valores de entrada de los modelos para especificar un rango de valores generalmente [0….1]. La normalización es principalmente usada en redes neuronales, modelos de regresión y clustering.
- **Discretización:** La discretización de los campos numéricos de entrada son modelados desde valores continuos a valores discretos usando intervalos.
- **Modelado de valores:** cualquier valor discreto puede ser modelado a cualquier posible valor discreto diferente, listando las parejas de valores.
- **Funciones:** obtiene valores aplicando una función a unos o más parámetros.
- **Agregación:** resume o recolecta grupos de valores.

2.5.5 Extensión

Este elemento debe estar presente como primer elemento o grupo de elementos definido en PMML, de esta manera es posible poner la información en los elementos de extensión

Name: Identifica dicha extensión en el documento PMML

Value: contiene el valor específico de la extensión

Estructura del elemento Extensión:

```
<xs:element name="Extension">
  <xs:complexType>
   <xs:complexContent mixed="true">
    <xs:restriction base="xs:anyType">
     <xs:sequence>
      <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="extender" type="xs:string" use="optional"/>
     <xs:attribute name="name" type="xs:string" use="optional"/>
     <xs:attribute name="value" type="xs:string" use="optional"/>
    </xs:restriction>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
```

2.5.6 MINING SCHEMA

Cada modelo contiene un esquema de minería, que enumera los campos usados en el modelo. Estos campos son un subconjunto de los campos en el diccionario de los datos. *Mining Schema* contiene la información que es específica a cierto modelo, mientras que *Data Dictionary* contiene las definiciones de los datos que no varían con el modelo.

2.5.7 MINIG-FUNTION

Ciertos tipos de modelos de PMML como redes neuronales y clustering son usados con diferentes propósitos, es decir, algunas son usadas para la predicción de valores numéricos y otras son usadas para la clasificación. PMML define cinco tipos de funciones de minería. Este modelo contiene un atributo ***FuntionName*** que especifica el nombre de la función de minería.

Los cinco tipos de funciones de minería son:

- Reglas de asociación
- Secuencia

- Clasificación
- Regresión
- Clustering

Estructura de Mining-Funtion:

```
<xs:simpleType name="MINING-FUNCTION">
  <xs:restriction base="xs:string">
    <xs:enumeration value="associationRules"/>
    <xs:enumeration value="sequences"/>
    <xs:enumeration value="classification"/>
    <xs:enumeration value="regression"/>
    <xs:enumeration value="clustering"/>
  </xs:restriction>
</xs:simpleType>
```

2.5.8 TAXONOMÍAS Y JERARQUÍAS

Los valores de un campo específico se pueden organizar en una jerarquía. Los ejemplos notorios son grupos de producto y jerarquías geográficas tales como ciudad, región, estado, país. Las jerarquías también se conocen como taxonomías o diagrama de clasificación.

La representación de jerarquías en PMML se basa en relaciones de padre/hijo. Un formato tabular es usado para proporcionar los datos a estas relaciones.

Una taxonomía se construye de una secuencia de una o más tablas padre/hijo. Los valores reales se pueden almacenar en tablas externas, esta tabla es referida por un ***TableLocator****.* Los datos tabulares pueden también ser parte del documento PMML en este caso se utiliza el elemento ***InlineTable*** *en lugar de TableLocator.*

ChildField: Define el nombre del campo que contiene el valor del hijo a registrar.

ParentField: Define el nombre del campo que contiene el valor del padre a registrar.

ParentLevelField: Define el nombre del campo que contiene el número del nivel del padre. Es opcional porque los niveles pueden ser derivados de los datos de hijo/padre.

IsRecursive: Una tabla recurrente puede definir una taxonomía completa en una tabla. Es decir, un valor en el campo del padre se puede también utilizar en un campo del hijo.

Estructura de Taxonomía:

```
<xs:element name="Taxonomy">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
       <xs:element ref="ChildParent" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="name" type="xs:string" use="required"/>
   </xs:complexType>
 </xs:element>

 <xs:element name="ChildParent">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
       <xs:choice>
         <xs:element ref="TableLocator"/>
         <xs:element ref="InlineTable"/>
       </xs:choice>
     </xs:sequence>
     <xs:attribute name="childField" type="xs:string" use="required"/>
     <xs:attribute name="parentField" type="xs:string" use="required"/>
     <xs:attribute name="parentLevelField" type="xs:string" use="optional"/>
     <xs:attribute name="isRecursive" use="optional" default="no">
       <xs:simpleType>
         <xs:restriction base="xs:string">
           <xs:enumeration value="no"/>
           <xs:enumeration value="yes"/>
         </xs:restriction>
       </xs:simpleType>
     </xs:attribute>
   </xs:complexType>
 </xs:element>

 <xs:element name="TableLocator">
   <xs:complexType>
     <xs:sequence>
       <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
     </xs:sequence>
```

2.5.9 Model Verification

Los proveedores y consumidores de PMML necesitan un mecanismo para asegurar que el desarrollo de un modelo en un nuevo ambiente genere resultados consistentes con el ambiente donde el modelo fue desarrollado.

El esquema de ***ModelVerification*** proporciona un conjunto de datos de las entradas del modelo y los resultados conocidos que se pueden utilizar para verificar que los resultados correctos están generados, sin importar el ambiente.

Para utilizar *ModelVerification*, el productor de un modelo de PMML agrega un sistema de registros para la verificación del modelo. Estos registros deben contener tanto los casos normales como casos excepcionales. Los resultados de la verificación pueden tomar muchas formas tales como el abastecimiento de un resultado para cada expediente de la verificación o registrar un listado con los errores de la verificación.

Estructura de ModelVerification:

```
<xs:element name="ModelVerification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="VerificationFields" />
      <xs:element ref="InlineTable" />
    </xs:sequence>
    <xs:attribute name="recordCount" type="INT-NUMBER" use="optional"/>
    <xs:attribute name="fieldCount" type="INT-NUMBER" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="VerificationFields">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
      <xs:element maxOccurs="unbounded" ref="VerificationField" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="VerificationField">
 <xs:complexType>
    <xs:sequence>
      <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="field" type="xs:string" use="required" />
    <xs:attribute name="column" type="xs:string" use="optional" />
    <xs:attribute name="precision" type="xs:double" default="1E-6"/>
    <xs:attribute name="zeroThreshold" type="xs:double" default="1E-16"/>
  </xs:complexType>
</xs:element>
```

ModelVerification esta compuesto por dos elementos:

VerificationFields: contiene los elementos de VerificationField que aparecerán en el InlineTable. Cada VerificationField se refiere a un campo del *MiningSchema* o de un *OutputField* que no sea parte *del MiningSchema*.

InlineTable: compara los campos especificados en la sección de VerificationFields y las cualidades opcionales para indicar el número de los expedientes de la verificación y el número de campos en cada registro.

2.6 Descripción de modelos

2.6.1 Reglas de asociación

Un modelo de reglas de asociación representa las reglas donde un conjunto de elementos se asocia a otro grupo de elementos, por ejemplo una regla puede expresar que un cierto producto o grupo de productos será comprado a menudo con cierto grupo de otros productos. Un modelo de reglas de asociación consiste en cuatro partes importantes: la declaración de los atributos que aparecen en el modelo, los elementos de entrada, los ítems set y las reglas de asociación.

Los atributos que aparecen en el modelo se definen antes de la declaración del modelo tales como el número de transacciones, las cuales son usadas entre otras cosas para calcular la confianza de la regla, la cantidad de elementos que van a estar contenidas en el modelo, la cantidad de reglas que se obtienen, el número de itemsets que contiene el modelo.

Los objetos usados en el modelo son definidos en la etiqueta ***Item***. A cada objeto le es asignado un id por el cual es identificado y el valor de dicho objeto. Obviamente la identificación de cada elemento así como su valor deben ser únicos.

Los elemento que se van a relacionar son definidos la etiqueta ***itemsets,*** cada relación es identificada por un id el cual debe ser único en el modelo, el atributo “support” es una porción del número de objetos en los datos para lo cuales una secuencia se mantiene cierta con respecto al total de objetos en los datos, el elemento itemref contiene una identificación del objeto al cual se hace referencia.

Las reglas son definidas en el elemento ***AssociationRule, se definen atributos los cuales forman la regla en cuestión:***

antecedent: contiene el valor del id del itemset que es el antecedente de la regla.

consequent: El valor del id del itemset que es el consiguiente de la regla.

La regla se vería de la siguiente forma antecedent => consequent.

2.6.2 Regresión

Las funciones de regresión son utilizadas para determinar las relaciones entre una variable dependiente y una o más variables dependientes. La variable dependiente es aquella cuyo valor se quiere pronosticar, mientras que las variables independientes son aquellas variables en la que se basa la predicción. Mientras que el término *regression* usualmente se refiere a la predicción de valores numéricos, el elemento ***RegressionModel*** en PMML puede también ser usado para la *clasificación*. Esto se deba a que múltiples ecuaciones pueden ser combinadas en orden para predecir valores categóricos.

Si el atributo ***functionName*** es ***regresión*** entonces el modelo es usado para la predicción de valores numéricos en un dominio continuo. Este modelo puede contener exactamente una tabla de regresión.

Si el atributo ***functionName*** es ***classification*** entonces el modelo es usado para predecir una categoría. Este modelo puede contener exactamente una tabla de regresión para cada ***targetCategory.*** Los métodos de normalización describen cómo la predicción es convertida en un valor de confianza (también conocido como probabilidad).

RegressionTable o tabla de regresión es la tabla que lista los valores de las variables independientes. Si el modelo es usado para predecir campos numéricos, entonces esta es solo una tabla de regresión y el atributo ***targetCategory*** puede estar ausente. Si el modelo es usado para predecir campos categóricos, entonces son dos o más tablas de regresión y cada una puede tener el atributo ***targetCategory*** definido con un único valor.

2.6.3 Modelo de Clustering

PMML define modelos de clustering en dos clases diferentes, estos son modelos basados en centros y basados en distribuciones. Ambos modelos contienen el elemento **ClusteringModel** como etiqueta principal y comparten muchos otos tipos de elementos que son comunes para ambos modelos.

Un modelo de cluster básicamente consiste en una estructura de cluster o agrupamientos de valores que comparten una característica en común. Usando estos modelos podemos identificar agrupamientos relevantes en una estructura de datos disponible.

El modelo debe contener información sobre las medidas de distancia o similitud usadas para el agrupamiento de los datos que permite definir la congruencia de los cluster de acuerdo a un registro de entrada determinado. Este también puede contener información de toda la distribución de datos, tales como la matriz de convergencia u otras estadísticas.

Es importante definir cuales de los campos de la estructura de datos pueden ser usados para ejecutar el modelo de cluster. Los campos en el modelo de cluster o **ClusteringField,** permiten definir cómo dos valores pertenecientes al dominio deben ser confrontados. El peso que tiene un campo en el cálculo de la distancia o similitud total y otros valores depende del tipo de método que se use. En el elemento **CenterField** es definida la correspondencia entre los campos de entrada y sus coordenadas, los datos categóricos pueden tener más de una coordenada asociada a ellos dependiendo del método de normalización que se use.

Los nombres de los campos en **ClusteringField** y **CenterField** deben estar en consistencia con los nombres de los campos definidos en el DataDictionary el TransformationDictionary. Cada cluster en el modelo tiene un arreglo de datos numéricos que contiene los valores asociados a cada cluster, si algún método de normalización es definido para los campos de entrada entonces las coordenadas son definidas usando los valores normalizados.

2.6.4 Modelos de reglas de secuencia

Un modelo de reglas de secuencia representa reglas para varias estructuras o ítems. Por ejemplo una regla puede expresar que después de comprar productos A y B, los consumidores tienden a comprar el producto C tarde o temprano.

Los elementos fundamentales que componen este modelo son <**Sequence**> y <**SequenceRule**>. La primera estructura permite definir los elementos que pertenecen a una secuencia y los atributos que la caracterizan, como el número de elementos, el support (porción del número de

objetos en los datos para lo cuales una secuencia se mantiene cierta con respecto al total de objetos en los datos), el ID de la secuencia etc. La segunda estructura consiste en una secuencia antecedente (**AntecedentSequence**) y en una secuencia consiguiente (**ConsequentSequence**) separado por un delimitador que en este caso seria un elemento de tiempo que provee estadísticas sobre el tiempo transcurrido entre la secuencia antecedente y la consiguiente. Además en este elemento se define la confianza de la regla o **confidence,** que no es mas que la probabilidad de que la secuencia antecedente definida por una regla de secuencia este seguida de la consiguiente.

2.7 Scalable Vector Graphics (SVG)

SVG es un lenguaje para describir objetos gráficos de dos dimensiones. Este trata tres tipos de objetos gráficos, formas vectoriales (curvas y rectas), imágenes y texto. Estos objetos gráficos pueden ser agrupados, transformados, formateados y compuestos en objetos previamente creados. Los gráficos SVG pueden ser interactivos y dinámicos. (W3C September 2001)
Dibuja gráficas complejas y por otro lado, es posible transformar estos elementos interactivos. Para visualizar cualquier gráfico en formato SVG, es necesario descargar un *plug-in*[2], en este caso SVGView, lanzado por la empresa IBM y distribuido gratuitamente por la empresa Adobe Systems Incorporated, que proporciona a los desarrolladores una manera de ver las imágenes vectoriales con el formato estándar SVG.

.

2.8 Significado de SVG

Su nombre se debe a:

- **Scalable**: Los gráficos SVG pueden ser mostrados con diferentes resoluciones.

[2] *Plug-in*: término en ingles, se refiere a un pequeño programa o módulo, que necesita ser instalado en la computadora, permite dotar a una determinada aplicación de nuevas funciones, en este caso un navegador de páginas de Internet.

- **Vector**: Los gráficos tienen formato vectorial, líneas y curvas. Frente a los formatos raster (matriz de puntos) que almacenan información por cada píxel del gráfico.
- **Graphics**: Proporciona la sintaxis para describir gráficos usando XML.

2.9 MODELOS BÁSICOS

Rectángulos (incluye esquinas redondeadas opcionales): Define un rectángulo el un eje de coordenadas “X” y “Y” mediante la etiqueta “rec” la cual tiene una serie de atributos entre los que se pueden definir el ancho, largo, posición, color de relleno. Los bordes redondeados pueden ser alcanzados con los atributos “ry” y “rx”.

Círculos: Define un círculo basado en un punto que sería el centro del círculo y un radio[3], se define como una etiqueta “circle”, en la cual se especifican las coordenadas (x,y) del punto del centro y el largo del radio.

Elipses: Define una elipse en un sistema de coordenadas (x,y) basado en un punto de centro y dos radios, se define como una etiqueta “ellipse” en la cual se especifican en punto centro y el largo del radio en el eje de las “x” y en el eje de las “y”.

Líneas: Define un segmento de línea que comienza en un punto y termina en otro, está definida mediante la etiqueta “line” en la cual se especifican el punto de inicio y el de final así como el ancho de la línea, color de fondo.

Polilíneas: Define un grupo de líneas conectadas entre sí, Típicamente, el elemento polilínea define formas abiertas. Se define por la etiqueta “polyline”, contiene una lista de puntos que forman la figura.

[3] Radio: es cualquier segmento que va desde el centro a cualquier punto de la circunferencia.

Polígonos: Define un grupo de líneas conectadas entre sí, Típicamente, el elemento polígono a diferencia de la polilínea define formas cerradas. Se define por la etiqueta “polygon”, contiene una lista de puntos que forman la figura.

2.10 ESTRUCTURA DE UN DOCUMENTO SVG

Un documento SVG consiste en la combinación de varios elementos SVG contenidos dentro de una etiqueta “svg”. Un documento SVG puede también estar contenido dentro de otros archivos, por ejemplo con XML (es importante el uso de los espacios de nombres para indicar que las etiquetas “svg” y “ellipse” son del espacio de nombres de SVG):

```
<?xml version="1.0" standalone="yes"?>
<parent xmlns="http://example.org/"
     xmlns:svg="http://www.w3.org/2000/svg">
  <!– parent contents here –>
  <svg:svg width="4cm" height="8cm" version="1.1">
    <svg:ellipse cx="2cm" cy="4cm" rx="2cm" ry="1cm" />
  </svg:svg>
  <!– ... –>
</parent>
```

2.9.1 ELEMENTO “SVG”

El elemento “svg” puede aparecer en la mitad del contenido de un SVG. Este es el mecanismo mediante el cual los fragmentos de documento SVG pueden ser incluidos dentro de otros fragmentos de documento SVG. Otro uso de los elementos “svg” en la mitad de un SVG es para establecer un nuevo marco llamado viewport.

Atributos del elemento “svg”

baseProfile = profile-name: Describe el mínimo perfil de lenguaje SVG que el autor cree que es necesario para una correcta visualización del SVG. El atributo no especifica ninguna restricción de proceso, puede ser considerado como un metadato.

x = "<coordinate>": La coordenada del eje x de la esquina de una región rectangular en la que está incrustado el elemento "*svg*". Si no es especificado el atributo, tomará el valor por defecto que es 0.

y = "<coordinate>": La coordenada del eje y de la esquina de una región rectangular en la que está incrustado el elemento "*svg*". Si no es especificado el atributo, tomará el valor por defecto que es 0.

width = "<length>": Para los elementos fuera del "*svg*", es la anchura del fragmento del documento SVG. Para los elementos incrustados en el "*svg*", es la anchura de la región en la cual el elemento "*svg*" está situado. Un valor negativo es un error y un valor 0 evita que el SVG se procese.

height = "<length>": Para los elementos fuera del "*svg*", es la altura del fragmento del documento SVG. Para los elementos incrustados en el "*svg*", es la altura de la región en la cual el elemento "*svg*" está situado. Un valor negativo es un error y un valor 0 evita que el SVG se procese.

viewBox: Si un documento SVG quiere ser usado como componente en otro documento, es frecuente que el autor quiera incluir un atributo *viewBox* en el elemento "*svg*" externo del documento referenciado. Este atributo provee una forma sencilla de diseñar documentos SVG para escalarse y ajustarse a un marco (*viewport*) arbitrario.

2.10.2 Grupo: elemento "g"

El elemento “g” es un contenedor para agrupar los elementos gráficos referenciados. La construcción de grupos, cuando son usadas con los elementos “desc” y ”title” proveen información adicional sobre la estructura y la semántica de un documento.

Un grupo de elementos, al igual que los objetos individuales, pueden ser nombrados usando el atributo “id”. Los nombres de los grupos son usados para diversos propósitos cómo la animación o la reusabilidad de los objetos. Por ejemplo:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="5cm" height="5cm" version="1.1"
    xmlns="http://www.w3.org/2000/svg">
 <desc>Two groups, each of two rectangles
 </desc>
 <g id="group1" fill="red" >
  <rect x="1cm" y="1cm" width="1cm" height="1cm" />
  <rect x="3cm" y="1cm" width="1cm" height="1cm" />
 </g>
 <g id="group2" fill="blue" >
  <rect x="1cm" y="3cm" width="1cm" height="1cm" />
  <rect x="3cm" y="3cm" width="1cm" height="1cm" />
 </g>
<!– Show outline of canvas using 'rect' element –>
 <rect x=".01cm" y=".01cm" width="4.98cm" height="4.98cm"
      fill="none" stroke="blue" stroke-width=".02cm" />
</svg>
```

Un elemento “g” puede contener otros elementos “g” anidados, con una profundidad arbitraria. Como se ve en el siguiente ejemplo:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="4in" height="3in" version="1.1"
    xmlns="http://www.w3.org/2000/svg">
  <desc>Groups can nest
  </desc>
  <g>
    <g>
      <g>
      </g>
    </g>
  </g>
</svg>
```

2.10.3 REFERENCIAS: EL ELEMENTO "DEFS"

SVG permite el uso de referencias URL a otros objetos. Por ejemplo, para rellenar un rectángulo con gradientes lineales, primero hay que definir un elemento "linearGradient"' y darle un ID:

<linearGradient id="MyGradient">…</linearGradient>

Después hay que usar la referencia al gradiente lineal como valor para la propiedad "fill" del rectángulo:

<rect style="fill:url(#MyGradient)"/>

Las referencias URI pueden ser definidas de las dos siguiente formas:

<URI-reference> = [<absoluteURI> | <relativeURI>] ["#" <elementID>]

<URI-reference> = [<absoluteURI> | <relativeURI>] ["#xpointer(id(" <elementID> "))"]

donde <elementID> es el ID del elemento referenciado.

El elemento "defs" es un contenedor para elementos que son referenciados. Por razones de entendimiento y accesibilidad, es recomendable que siempre que sea posible, los elementos referenciados sean definidos dentro de un elemento "defs".

El modelo de contenido de "defs" es el mismo que para el elemento "g". Sin embargo, los elementos de "defs" no serán visualizados directamente, si no tendrán que esperar a ser

referenciados, este funcionamiento es idéntico al de un elemento “g” con la propiedad “display” igual a “none”.

2.10.4 ELEMENTO “USE”

Todo elemento “svg”, “symbol”,”g”, elementos gráficos u otro “use” es potencialmente un objeto plantilla que puede ser re-usado o “instanciado” en un documento SVG usando el elemento “use”. El elemento “use” hace referencia a otro elemento e indica que el contenido gráfico de ese elemento es incluido/dibujado en el momento indicado en el documento. Al contrario del elemento “image”, el elemento “use” no puede referenciar ficheros enteros.

El elemento “use” tiene varios atributos opcionales (x, y, width, height) que pueden ser usados para posicionar el elemento referenciado en una región rectangular según el sistema actual de referencias.

El efecto del elemento “use” es como si los elementos referenciados con todos sus descendientes fueran clonados en el árbol del documento XML justo donde estaba el elemento “use”. A continuación hay un pequeño ejemplo del uso del elemento “use” con un “rect”.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="10cm" height="3cm" viewBox="0 0 100 30" version="1.1"
    xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
 <desc>Example Use01 - Simple case of 'use' on a 'rect'</desc>
 <defs>
  <rect id="MyRect" width="60" height="10"/>
 </defs>
 <rect x=".1" y=".1" width="99.8" height="29.8"
      fill="none" stroke="blue" stroke-width=".2" />
 <use x="20" y="10" xlink:href="#MyRect" mce_href="#MyRect" />
</svg>
```

2.10 Conclusiones del Capítulo

Lo más interesante a corto y medio plazo con la adopción de estos estándares es que, a medida que aplicaciones en el mercado ofrezcan la posibilidad de obtener las salidas de sus modelos en formato XML, esto permitirá a distintas aplicaciones, en distintos entornos o plataformas, e incluso bajo distintos Sistemas Operativos llegar a ser más o menos independientes de la herramienta usada para visualizar el modelo. PMML servirá, de este modo, de puente como interfaz entre un sistema de minería de datos y el amplio universo de aplicaciones que necesitan de los resultados de sus procesos. SVG por su parte nos brinda un camino para describir estos modelos en forma gráfica, es por eso que en el siguiente capitulo esta encaminado a proporcionar un método de transformación de documentos PMML a SVG.

Capítulo 3 Caso de estudio

3.1 Introducción al Capitulo

Con la utilización de SVG es posible visualizar de forma gráfica cualquier tipo de documento XML en este caso un PMML el cual contiene el resultado de un modelo de minería. En este capítulo se presenta una vía para trasformar un PMML en un SVG mediante la utilización de otro estándar conocido como XSLT, utilizado para la transformación de documentos XML. Se agrega un caso de estudio de un documento PMML que muestra el resultado de un algoritmo de clustering el cual es trasformado mediante un XSLT y convertido a un documento SVG para mostrar la información almacenada de forma gráfica.

3.2 Transformar un documento PMML mediante un XSLT

Para transformar un documento PMML a SVG proponemos la utilización de un lenguaje etiquetado conocido como XSLT (eXtensible Stylesheet Language Transformation), el cual es un estándar de la organización W3C basado en XML. XSLT es un lenguaje para trasformar la estructura de un documento XML.(KAY August 2004) Puede trasformar un documento XML en otros e incluso en formatos que no son XML.

El lenguaje XPath forma una parte esencial de XSLT, aunque se define realmente en una recomendación separada de W3C, puede también ser utilizada independientemente de XSLT. XPath hace referencia a la forma de acceder y moverse sobre la estructura de un documento XML. (W3C January 2007)

XSLT realiza la transformación del documento utilizando una o varias reglas, unidas al documento fuente a trasformar y mediante un procesador XSLT es posible obtener un archivo de salida en formato SVG con las trasformaciones deseadas.

3.3 ELEMENTOS XSLT

xsl:apply-imports: El elemento <xsl:apply-imports> es complejo en su uso, y es utilizado mayoritariamente en hojas de estilo muy complejas. La precedencia de importación indica que las plantillas en la hoja de estilo principal tienen mayor precedencia que las plantillas en las hojas de estilo importadas. Sin embargo, en ocasiones es útil forzar al procesador para que aplique una plantilla de menor procedencia contenida en la hoja de estilo importada en lugar de una plantilla equivalente en la hoja de estilo principal.

xsl:apply-templates: El elemento <xsl:apply-templates> selecciona un conjunto de nodos del documento de entrada e instruye al procesador para aplicar las plantillas apropiadas a ellos.

xsl:attribute: El elemento <xsl:attribute> define un nuevo atributo que puede ser utilizado en cualquier parte del documento de entrada o de salida, a este atributo se le define un nombre por el cual es identificado y un valor.

xsl:call-template: El elemento <xsl:call-template> invoca una plantilla con nombre.

xsl:cose: El elemento <xsl:choose> define una elección entre un número de alternativas. Funciona como una sentencia switch en otros lenguajes de programación.

xsl:decimal-format: El elemento <xsl:decimal-format> define los caracteres y los símbolos que serán usados en la conversión de números a cadenas de texto usando la función format-number.

xsl:for-each: El elemento <xsl:for-each> selecciona un conjunto de nodos y procesa cada uno de ellos de la misma manera. Se utiliza fundamentalmente para iterar a través de un conjunto de

nodos o para cambiar el nodo actual. Si se encuentran uno o más elementos <xsl:sort> como hijos de este elemento, los nodos se ordenan antes del procesamiento. De otra manera, los nodos se procesarán en el orden del documento.

xsl:if: El elemento <xsl:if> contiene un atributo a probar y una plantilla. Si el atributo resulta verdadero, la plantilla es procesada. Este comportamiento es similar a la sentencia if de otros lenguajes. Sin embargo, para conseguir la funcionalidad de una sentencia if-then-else, es necesario utilizar el elemento <xsl:choose> con un elemento hijo <xsl:when>, y otro elemento hijo <xsl:otherwise>

3.4 CASO DE ESTUDIO

Como ya vimos anteriormente el SVG es un vocabulario XML para la definición de gráficos vectoriales. Esto significa por tanto que para la edición, manipulación y generación de gráficos vectoriales con este vocabulario podemos utilizar las herramientas que normalmente utilizamos para trabajar con cualquier documento XML.

El siguiente documento PMML muestra la salida de un algoritmo de clustering aplicado a una colección de datos referente a un grupo de personas donde se muestra la edad, salario y estado marital que puede ser soltero o casado. Este documento almacena en arreglos (tantos como sean necesarios) el resultado de un algoritmo de clustering.

```
<?xml version="1.0" ?>
<PMML version="3.1" xmlns="http://www.dmg.org/PMML-3_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Header copyright="dmg.org"/>
  <DataDictionary numberOfFields="3">
   <DataField name="marital status" optype="categorical" dataType="string">
      <Value value="s"/>
      <Value value="d"/>
```

```
      <Value value="m"/>
    </DataField>
    <DataField name="age" optype="continuous" dataType="double"/>
    <DataField name="salary" optype="continuous" dataType="double"/>
  </DataDictionary>
  <ClusteringModel modelName="Mini Clustering"
      functionName="clustering"
      modelClass="centerBased"
      numberOfClusters="2">
    <MiningSchema>
          <MiningField name="marital status"/>
          <MiningField name="age"/>
          <MiningField name="salary"/>
    </MiningSchema>

    <ComparisonMeasure kind="distance">
          <squaredEuclidean/>
    </ComparisonMeasure>

    <ClusteringField field="marital status"
                compareFunction="absDiff"/>
    <ClusteringField field="age"
                compareFunction="absDiff"/>
    <ClusteringField field="salary"
                compareFunction="absDiff"/>
    <CenterFields>
      <DerivedField name="c1" optype="continuous" dataType="double">
        <NormContinuous field="age">
          <LinearNorm orig="45" norm="0"/>
          <LinearNorm orig="82" norm="0.5"/>
          <LinearNorm orig="105" norm="1"/>
```

```
      </NormContinuous>
    </DerivedField>
    <DerivedField name="c2" optype="continuous" dataType="double">
      <NormContinuous field="salary">
          <LinearNorm orig="39000" norm="0"/>
          <LinearNorm orig="39800" norm="0.5"/>
          <LinearNorm orig="41000" norm="1"/>
      </NormContinuous>
    </DerivedField>
    <DerivedField name="c3" optype="continuous" dataType="double">
      <NormDiscrete field="marital status" value="m"/>
    </DerivedField>
    <DerivedField name="c4" optype="continuous" dataType="double">
      <NormDiscrete field="marital status" value="d"/>
    </DerivedField>
    <DerivedField name="c5" optype="continuous" dataType="double">
      <NormDiscrete field="marital status" value="s"/>
    </DerivedField>
  </CenterFields>
  <MissingValueWeights>
    <Array n="5" type="real">1 1 1 1 1</Array>
  </MissingValueWeights>
  <Cluster name="marital status is d or s">
    <Array n="5" type="real">
          0.384561 0.306321 0.238427 0.199188 0.332384</Array>
  </Cluster>
  <Cluster name="marital status is m">
    <Array n="5" type="real">
           0.69946 0.609037 0.73226 0.783521 0.655253</Array>
  </Cluster>
</ClusteringModel>
```

```
</PMML>
```

A este PMML se le aplicó un conjunto de reglas XSLT, lenguaje que también vimos anteriormente en este capitulo para extraer la relación que existe entre personas según su estado marital teniendo en cuenta su edad y salario. Se obtiene como salida un documento SVG con la información gráfica referente a la salida del algoritmo de clustering.

Generando la información en formato SVG

En este apartado vamos a mostrar el documento PMML anterior en formato SVG de manera que podamos visualizar la información que proporciona de forma gráfica.

Al ser el SVG un vocabulario XML podemos realizar esta conversión utilizando el lenguaje de transformación XSLT y cualquier procesador XSLT estándar (yo he utilizado el XML SPY).

El XSLT comienza con la definición de una serie de variables en las que se definen y calculan aspectos como el tamaño de la ventana, gráfica, leyenda y los colores para cada uno de los cluster que serán representados:

```
<!-- Tamaño de la ventana -->
    <xsl:variable name="Ventana">400</xsl:variable>
    <xsl:variable name="Margin">50</xsl:variable>

    <!-- Tamaño de la gráfica -->
    <xsl:variable name="Grafica">
        <xsl:value-of select="$Ventana - $Margin*2"/>
    </xsl:variable>

    <!-- Tamaño de la leyenda -->
    <xsl:variable name="Width">10</xsl:variable>
    <xsl:variable name="Height">7</xsl:variable>
    <xsl:variable name="textLength">60</xsl:variable>
    <xsl:variable name="interval">10</xsl:variable>
```

```
    <xsl:variable name="pos">0</xsl:variable>

<!--Definimos lo colores para cada cluster-->
    <xsl:variable name="colors">
        <xsl:element name="color">red</xsl:element>
        <xsl:element name="color">blue</xsl:element>
        <xsl:element name="color">black</xsl:element>
        <xsl:element name="color">green</xsl:element>
    </xsl:variable>
```

En esta regla de construcción recorriendo los cluster tag y obteniendo el valor de Array tag, y lo almacenamos en la variable "value" a la que le aplicamos la función split y devuelve el resultado en forma de node set en la variable "tokens":

```
<xsl:template name="cluster">
    <xsl:for-each select="pmml:Cluster">
      <xsl:value-of select="$pos = $pos + 1"/>
      <xsl:variable name="value" select="pmml:Array"/>
      <xsl:variable name="tokens">
        <xsl:call-template name="str:split">
            <xsl:with-param name="string" select="$value"/>
            <xsl:with-param name="pattern" select="' '"/>
        </xsl:call-template>
      </xsl:variable>
```

Y luego pintamos los puntos en la gráfica

```
<g id="bar" transform="translate(0,200)">
      <xsl:for-each select="$tokens/token">
          <xsl:variable name="XY" select="."/>
          <rect x="{$XY * $Grafica + $Margin}" y="-{$XY*$Grafica + $Margin + (400 - $Ventana)}"
fill="{$colors/color[$pos]}" width="0.5" height="0.5" stroke="{$colors/color[$pos]}" stroke-
width="2"/>
      </xsl:for-each>
      <xsl:call-template name="legend">
          <xsl:with-param name="text" select="@name"/>
          <xsl:with-param name="colorValue" select="$colors/color[$pos]"/>
          <xsl:with-param name="intervalValue" select="$interval"/>
      </xsl:call-template>
```

```
        <xsl:value-of select="$interval = $interval + $interval"/>
  </g>
    </xsl:for-each>
</xsl:template>
```

En esta regla de construcción pintamos los ejes de coordenadas:

```
<xsl:template name="coordenadas">
        <!-- Pintamos la coordenada Y -->
        <line x1="{$Margin}" y1="-{200 - $Margin}" x2="{$Margin}" y2="{(-200 + $Margin) + $Grafica}"
style="stroke:#000000; stroke-width:1"/>
        <!-- Test Y -->
        <text x="{$Margin - 30}" y="-{200 - $Margin - 10}" fill="red">Edad</text>
        <!-- Pintamos la coordenada X -->
        <line x1="{$Margin}" y1="{(-200 + $Margin) + $Grafica}" x2="{$Ventana - $Margin}" y2="{(-200
+ $Margin) + $Grafica}" style="stroke:#000000; stroke-width:1"/>
        <!-- Test X -->
        <text x="{$Ventana - $Margin - 30}" y="{(-200 + $Margin) + $Grafica + 15}"
fill="red">Salario</text>
        <xsl:call-template name="lhorizontal"/>
    </xsl:template>
```

En esta regla de construcción pintamos las líneas horizontales:

```
<xsl:template name="lhorizontal">
        <line x1="{$Margin}" y1="{(-200 + $Margin) + $Grafica*0.25}" x2="{$Ventana - $Margin}"
y2="{(-200 + $Margin) + $Grafica*0.25}" style="stroke:#000000; stroke-width:0.1"/>
        <text x="{$Margin div 2 - 5}" y="{(-200 + $Margin) + $Grafica*0.25}" fill="red">0.75</text>
        <line x1="{$Margin}" y1="{(-200 + $Margin) + $Grafica*0.50}" x2="{$Ventana - $Margin}"
y2="{(-200 + $Margin) + $Grafica*0.50}" style="stroke:#000000; stroke-width:0.1"/>
        <text x="{$Margin div 2 - 5}" y="{(-200 + $Margin) + $Grafica*0.50}" fill="red">0.50</text>
        <line x1="{$Margin}" y1="{(-200 + $Margin) + $Grafica*0.75}" x2="{$Ventana - $Margin}"
y2="{(-200 + $Margin) + $Grafica*0.75}" style="stroke:#000000; stroke-width:0.1"/>
        <text x="{$Margin div 2 - 5}" y="{(-200 + $Margin) + $Grafica*0.75}" fill="red">0.25</text>
        <text x="{$Margin div 2 - 5}" y="{(-200 + $Margin) + $Grafica}" fill="red">0</text>
    </xsl:template>
```

En esta regla de construcción pintamos la leyenda:

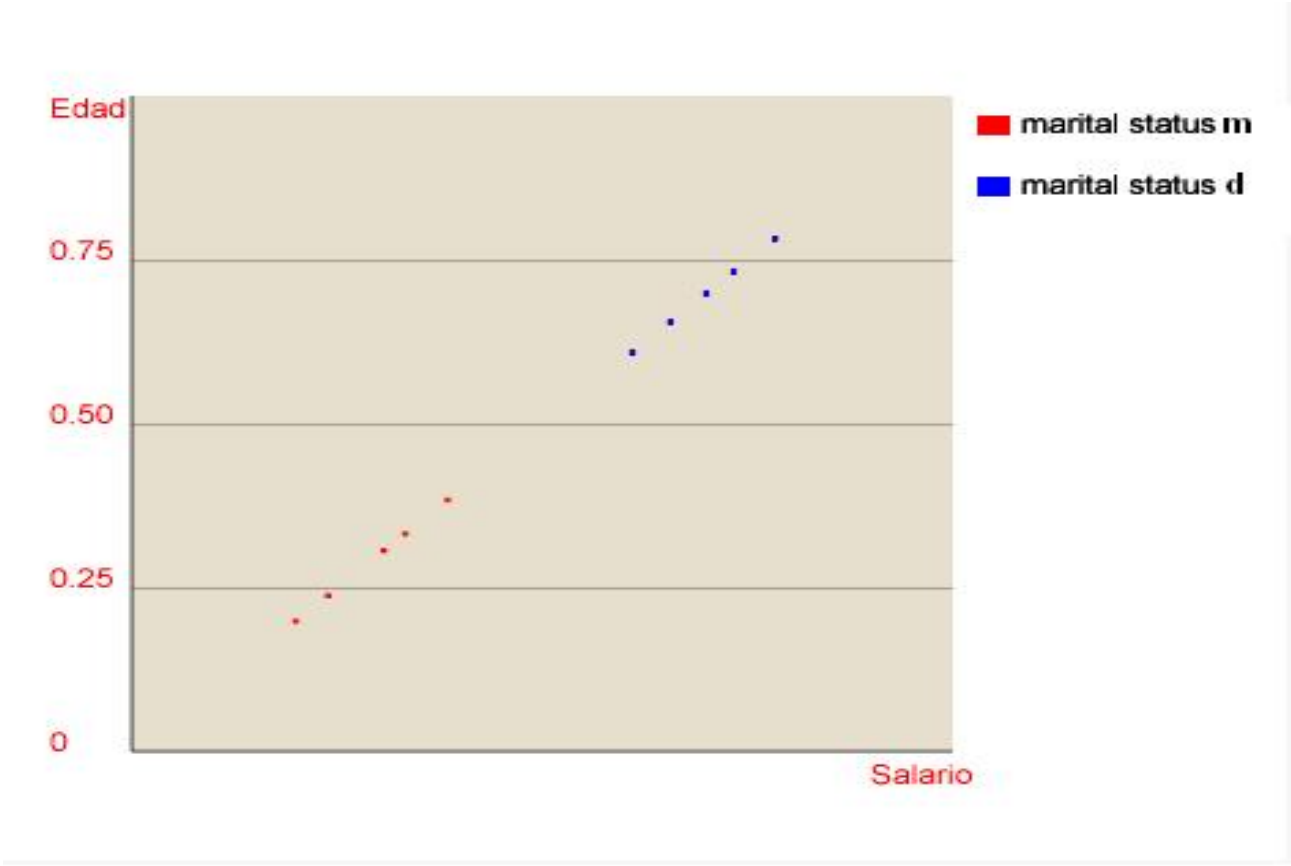
```
<xsl:template name="legend">
        <xsl:param name="text"/>
        <xsl:param name="colorValue"/>
```

```
        <xsl:param name="intervalValue"/>
        <rect x="{$Ventana - $Margin + 10}" y="-{400 - $Margin - $intervalValue}" fill="{$colorValue}"
width="{$Width}" height="{$Height}" stroke="{$colorValue}" stroke-width="2"/>

        <text x="{$Ventana - $Margin + 15 + $Width}" y="-{400 - $Margin - $intervalValue - $Height}"
fill="black"><xsl:value-of select="$text"/></text>

    </xsl:template>
```

Y por último pintamos toda la gráfica aplicando las reglas de construcción anteriores:

```
<xsl:template match="pmml:ClusteringModel">
        <svg width="{$Ventana + $Width + $textLength}" height="{$Ventana}">
            <g id="bar" transform="translate(0,200)">
                <rect x="{$Margin}" y="-{200 - $Margin}" width="{$Grafica}" height="{$Grafica}"
style="fill:                                    #E3DFCC;"/>
                <xsl:call-template name="coordenadas"/>
                <xsl:call-template name="cluster"/>
            </g>
        </svg>
    </xsl:template>
```

Y finalmente obtenemos el resultado gráfico de la transformación, para la visualización de este es necesario instalar en SVG Viewer de Adobe ofrece de forma gratuita.


Edad
0.75
0.50
0.25
0
Salario
marital status m
marital status d

CONCLUSIONES

En el desarrollo de este trabajo se realizó un estudio acerca de las técnicas de visualización del conocimiento, además de los estándares vinculados a este campo. Las técnicas de visualización son de vital importancia para la toma de decisiones empresariales, se basan en representaciones gráficas para visualizar la información de forma que sea entendible por aquellos que necesitan utilizarla.

Durante el estudio de esta investigación surgen formas novedosas de resolver el problema mediante estándares abiertos, los cuales han tomado vital importancia en los últimos años. Los estándares vinculados a este tema son PMML utilizado para describir modelos predictivos, SVG describe objetos gráficos en dos dimensiones y el estándar XSLT utilizado para la transformación de documentos XML.

Es necesario destacar que con este trabajo se alcanza la realización del objetivo propuesto inicialmente en el comienzo de esta investigación, de proponer un mecanismo de visualización de los resultados del análisis inteligente de datos y de esta forma facilitar que sea entendible para aquellas personas que necesitan de esta información para la toma de decisiones.

RECOMENDACIONES

Representar la información de forma gráfica es de vital importancia para aquellas personas que necesitan validar sus decisiones, es por esto que se recomienda seguir trabajando en la confección de XSLT para los restantes modelos predictivos que pueden estar contenidos en un documento PMML y este pueda ser trasformado a formato SVG y ser visualizado de forma gráfica.

Seguir investigando sobre los estándares existentes para la visualización de la información

Seguir investigando en otras posibles vías para trasformar un PMML en una representación visual.

Recomendamos la utilización de este mecanismo de transformación en los proyectos investigativos de la universidad así como por la dirección del proyecto UCI como método para visualizar la información.

REFERENCIAS BIBLIOGRÁFICAS

1. DAVIDSON, T. S. A. I. *Visual Data Mining: Techniques and Tools for Data Visualization and Mining*. John Wiley & Sons Inc, May 2002. 382 pages p. ISBN: 0471149993
2. GROUP, D. M. *PMML*. Disponible en: http://www.dmg.org/
3. JOSÉ ANTONIO LÓPEZ PÉREZ. *Integración de Aplicaciones en Data Mining: XML y PMML*, abril 2002.
4. JOSEPH NOVAK, G., D. BOB. *APRENDIENDO A APRENDER*. 1984. p. 0-521-31926-9
5. MERCEDES VITTURINI, K. C., SILVIA CASTRO. *VISUALIZACIÓN DE GRANDES VOLÚMENES DE DATOS*. Departamento de Ciencias de la Computación. Bahía Blanca, Universidad Nacional del Sur - Bahía Blanca. p.
6. *XML Path Language (XPath) 2.0*, January 2007. [Disponible en: http://www.w3.org/TR/2007/REC-xpath20-20070123/
7. KAY, M. *XSLT 2.0 Programmer's Reference*. August 2004. p. ISBN: 978-0-7645-6909-8
8. W3C. *Scalable Vector Graphics (SVG) 1.0 Specification*, September 2001. [Disponible en: http://www.w3.org/TR/2001/REC-SVG-20010904/
9. AMECE, P. P. L. E. *Estándares*, 2007. [Disponible en: http://www.amece.org.mx/amece/oc/content.php?id=15

BIBLIOGRAFÍA CONSULTADA

1. CHRISTOPH LINGENFELDER, S. R. Explanation of PMML Models, 2005.
2. COVER, R. *New Release of Predictive Model Markup Language (PMML) from SourceForge.*, mayo 2005. [Disponible en: http://xml.coverpages.org/ni2005-05-27-a.html
3. *Predictive Model Markup Language (PMML)*, mayo 2005. [Disponible en: http://xml.coverpages.org/pmml.html
4. GROSSMAN, R. 2nd International Workshop on Data Mining Standards, Services and Platforms, 2004.
5. Proceedings of the First Annual Workshop on Data Mining Standards, Services, and Platforms, 2003.
6. GROSSMAN, R. L. PMML Models for Detecting Changes, 2005.
7. JOSÉ ANTONIO MOREIRO GONZÁLEZ, D. G. M. La visualización de la información en revistas electrónicas mediante la concurrencia de herramientas hipertextuales, mapas conceptuales, topic maps y ontologías.
8. LUIGI, Q. *Predictive Model Markup Language (PMML)*, 2001. p.
9. R. CORTÁZAR, M. V. L., J. OLIVER. *Métodos avanzados de presentación de datos*. 2001. p. I.S.B.N.: 84-7800-874-8
10. REYES, M. A. H. *Entorno de evaluación visual interactiva de modelos de reglas de asociación*. Madrid, Universidad politecnica de Madrid, 2006. p.
11. RÍO, A. C. D. *XSLT / XPath*. Departamento de Informática, Universidad de Oviedo. p.
12. ROBERT GROSSMAN, S. B., PHILIP HALLSTROM The Management and Mining of Multiple Predictive Models Using the Predictive Modeling Markup Language (PMML), 2007.
13. ROBERT L. GROSSMAN, M. F. H., GREGOR MEYER *Data mining standards initiatives*, 2002 [Disponible en: http://www.teraflowtestbed.net/publication_files/journal-018.htm
14. SARAB ANAND, M. G., DIETRICH WETTSCHERECK *ECML/PKDD-2003 Knowledge Discovery Standards* September 2003. p.
15. VILLATE, J. E. *Introducción al XML*, Universidad de Oporto, mayo de 2001. p.
16. W3C. *XSLT RequirementsVersion 2.0,* 2001.http://www.w3.org/TR/2001/WD-xslt20req-20010214

1