UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

DIRECCIÓN DE FORMACIÓN POSTGRADUADA

Aplicación informática que integra los procesos de generación, evaluación y uso de Clasificadores Bayesianos para dominios Bioinformáticos y Médicos.

Trabajo final presentado en opción al título de
Máster en Informática Aplicada

Autor: Ing. Elvismary Molina de Armas

Tutores: MSc. Mario Pupo Meriño,

MSc. Maikel Yelandi Leyva Vázquez

Consultante: Dra. María del Carmen Chávez Cárdenas

Junio 2011
Ciudad de La Habana, Cuba.

DECLARACIÓN DE AUTORÍA

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo, Elvismary Molina de Armas, con carné de identidad 85091806498, soy el autor principal del trabajo final de maestría “Implementación de una aplicación informática que integre los procesos de generación, evaluación y uso de Clasificadores Bayesianos para dominios Bioinformáticos y Médicos.”, desarrollada como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana a los ____ días del mes de _____ del año _____.

Ing. Elvismary Molina de Armas

_____

Firma del Autor

MSc. Mario Pupo Meriño

_____

Firma del Tutor

MSc. Maikel Y. Leyva Vázquez

_____

Firma del Tutor

RESUMEN

La presente investigación tiene como precedentes la necesidad de aplicar técnicas automatizadas de clasificación en los datos generados por los estudios realizados bajo el Programa del Sistema Nacional de Salud Pública en Cuba, y los resultados obtenidos por el Grupo de Bioinformática de la Universidad Central “Marta Abreu” de Las Villas (UCLV) para potenciar el uso de estas técnicas, en especial la Clasificación Bayesiana, con el desarrollo de nuevos algoritmos de aprendizaje estructural optimizados para dominios Bioinformáticos y Médicos.

En la misma, se muestran los resultados de la implementación de una aplicación informática basada en tecnología libre, que integra los procesos de generación de Clasificadores Bayesianos, incluyendo los algoritmos de aprendizaje estructural BayesChaid, ByNet y BayesPSO, la evaluación, tras la selección y verificación por parte del usuario de las métricas definidas para este tipo de estudio, y su uso en la clasificación de nuevos casos que puedan ser incluidos al conjunto inicial de entrenamiento para posteriores aprendizajes. Para alcanzar el objetivo anterior, se expone el estudio de las funcionalidades necesarias, y se realiza un diseño orientado hacia la extensión e integración de las funcionalidades de Weka.

Además, se presenta la validación del sistema con respeto a su correcto funcionamiento y el grado de satisfacción por parte del cliente, evidenciando que la utilización del mismo contribuirá en la disminución de la dificultad del uso de este tipo de modelo estadístico por parte de los especialistas.

PALABRAS CLAVES:

Clasificación
Clasificadores Bayesianos
Generación, evaluación y uso de los Clasificadores Bayesianos
Redes Bayesianas
Integración
Extensión de Weka
Integración con Weka

TABLA DE CONTENIDOS

INTRODUCCIÓN

En Cuba, como un elemento estratégico del Programa del Sistema Nacional de Salud Pública, se ha potenciado el desarrollo de la Genética Comunitaria y otras áreas de Atención Primaria a la Salud. Esto se materializa en estudios sobre sectores específicos de la población como son las personas con discapacidad física e intelectual y los gemelos, el programa materno infantil, el programa de enfermedades crónicas no transmisibles (HTA, DM, AB), el programa de cáncer bucal, entre otros, que generan grandes volúmenes de información. Igualmente, en el desarrollo de otras áreas de investigación relacionadas, como la Bioinformática y Biotecnología, se llevan a cabo estudios donde es necesario procesar grandes volúmenes de datos, por ejemplo en los desarrollados por el Grupo de Bioinformática de la UCLV y el de la UCI, y en los Centros del Polo Científico, como el CIM y el CIGB, haciéndose indispensable el uso de tecnologías de software para la generación de nuevo conocimiento a partir del análisis y procesamiento de los mismos.

Dentro de este grupo de investigaciones surgen con mucha frecuencia estudios en los cuales se quiere conocer el efecto de un conjunto de variables discretas sobre una variable que identifica la pertenencia a un grupo y que es supuestamente dependiente de ellas, teniendo por objetivo establecer una **clasificación** a determinada entidad definida por dichas variables. Por ejemplo, suele presentarse el caso en que a partir del conocimiento de los datos de un paciente, es necesario predecir su condición, a través de una clasificación que puede ser *enfermo* o *sano*, siendo esta variable (condición del paciente) supuestamente dependiente de un grupo de características (variables discretas predictoras) que pueden representar los riesgos o síntomas.

Además, este tipo de estudios se caracteriza por la existencia de fuentes masivas de datos, cuyo análisis completo resulta insostenible desde el punto de vista humano, y donde el porcentaje de redundancia y de datos ruidosos o sujetos a errores es muy elevado, lo cual ha obligado al uso de técnicas de análisis automatizadas que aprovechan las capacidades de cómputo del ordenador en pos de obtener patrones o modelos, posibilitando la generación de nuevo conocimiento a partir de los datos recopilados. Dentro de la amplia gama de software se encuentran los diseñados

específicamente para tareas de clasificación: Elvira [1], Hugin [2] y Netica [3], que al ser propietarios no cumplen con las políticas de informatización cubanas [4], y la plataforma Weka [5] y JavaBayes [6], desarrollados bajo tecnología libre.

Para entender mejor el análisis de las aplicaciones mencionadas anteriormente, es preciso antes detenerse en la presentación de los diferentes conceptos relacionados con la clasificación y su automatización:

La clasificación es una de las técnicas ancestrales más utilizadas en la comprensión del mundo. Se sustenta en el objetivo de establecer criterios que permitan la identificación, denominación y agrupamiento de las entidades que intervienen en un determinado dominio de estudio, evaluando similitudes en sus características o la relación establecida entre las variables que las definen. La misma está pautada por el grado de comprensión que se tenga de los diferentes renglones que definen a las entidades evaluadas, y forma parte de la generación de nuevo conocimiento, ya que permite sintetizar características comunes para definir funcionalidad, aplicación, diferentes tipos de enfoques, etc. Su aplicación se halla en todas las ramas del saber incluyendo la Biología y la Medicina, donde se encuentra presente en el dominio de las enfermedades, por ejemplo en su definición como *infecciosas, parasitarias, congénitas*, etc., o determinando el estado de un paciente, que puede ser clasificado como *enfermo* o *no enfermo.*

Dentro de las técnicas automatizadas de clasificación, se encuentran los Clasificadores Bayesianos. Estos permiten generar un modelo probabilístico que describe el dominio de estudio a partir de una fuente de datos de entrenamiento, teniendo en cuenta la incertidumbre presente en los mismos, y brindando la posibilidad de ser aplicados en la predicción de la probabilidad de que una muestra dada pertenezca a una clase particular. Los mismos utilizan como modelo matemático una Red Bayesiana (RB), definida por un grafo acíclico dirigido (GAD), donde se muestra la estructura relación-dependencia (arcos) entre las diferentes variables del dominio (nodos) y su distribución de probabilidad condicional.

La **generación** de una RB está determinada por dos tareas: el aprendizaje estructural,

que permite obtener la estructura de grafo a partir de las relaciones de dependencia condicional entre las variables, y el aprendizaje paramétrico, caracterizado por calcular la distribución de probabilidades (parámetros), lo que posibilitará realizar inferencias en la red. En relación a estas dos tareas, el aprendizaje estructural es esencial por ser realmente el más difícil e imprescindible para realizar el aprendizaje paramétrico, por lo cual es evidente, que las posibilidades del uso de una RB se fortalecen especialmente si se logra optimizar el aprendizaje estructural acorde con el dominio del campo de aplicación.

Además, existen diferentes técnicas de **evaluación** de una RB, definidas a partir de la relación establecida entre el conjunto de datos de entrenamiento y los de prueba (usar la propia base de datos de entrenamiento para pruebas, utilizar una base de datos independiente para pruebas, o utilizar un subconjunto o porcentaje de las mismas, de forma estática o alternada e iterativa (*validación cruzada*)), pudiendo ser aplicadas diferentes métricas en las mismas, como por ejemplo las definidas a partir de la matriz de confusión u otras de propósito general como la prueba de Friedman [7]. Finalmente, partiendo de los resultados de esta evaluación, el especialista puede realizar la selección del modelo que mejor represente el dominio de estudio, y pasar a la etapa de **uso** del clasificador bayesiano, utilizándolo para realizar inferencias sobre nuevas entidades. Por tanto el ciclo de vida de un clasificador bayesiano está integrado por tres procesos: **generación**, **evaluación** o validación y **uso**.

En el marco de las investigaciones vigentes en el país, mencionadas anteriormente, se llevan a cabo estudios para potenciar las técnicas de Clasificación, especialmente la Clasificación Bayesiana. Este es el caso específico del Grupo de Bioinformática de la Universidad Central “Marta Abreu” de Las Villas (UCLV), donde como parte del proceso de **generación** de los clasificadores bayesianos automatizados, se definieron y validaron tres nuevos algoritmos de aprendizaje estructural: *ByNet, BayesChaid, BayesPSO* [8], demostrándose su efectividad en dominios Bioinformáticos y Médicos, en base a su aplicación en la resolución de problemas de predicción de interacciones

de proteínas, sobre localización de splice-sites[1] y en predicción de la hipertensión arterial.

Para la validación práctica de estos nuevos algoritmos, después del estudio de las herramientas informáticas disponibles en el estado del arte, se usaron en una primera etapa productos de software tradicionales propietarios como Mathematica, SPSS, Microsoft Excel, etc. Posteriormente se realizó la implementación de los nuevos algoritmos como extensión de la plataforma de aprendizaje automatizado *Weka,* para la **generación** de los clasificadores bayesianos y parte de su **evaluación**, SPSS para la realización de otras pruebas de **validación** específicas como la prueba de Friedman, y para su **uso** se desarrolló de una aplicación informática denominada ByShell*,* que permitía realizar inferencias sobre los modelos generados, pero que no se encuentra validado totalmente y fue desarrollado utilizando la plataforma privativa Borlan Delphi.

Weka es una plataforma de código abierto de Minería de Datos que expone una extensa colección de algoritmos genéricos implementados en Java, útiles para ser aplicados mediante las interfaces que ofrece o para ser encapsulado dentro de cualquier aplicación. La misma contiene herramientas para realizar transformaciones sobre los datos, tareas de clasificación, regresión, agrupamiento, asociación y visualización. Sin embargo, dada su complejidad y su capacidad de ser genérico, es del criterio de ser una herramienta difícil de comprender y manejar, con usabilidad bastante pobre. Sumado a lo anterior, las potencialidades de Weka se expresan en la presentación de un marco para la generación y comparación de estas técnicas de aprendizaje automatizado, utilizando métricas de evaluación predefinidas y no seleccionables de forma independiente, pero una vez generado un modelo, específicamente un clasificador bayesiano, no permite asignar una clasificación a un nuevo caso ni su permanencia posterior en el conjunto de estudio, limitando su uso solo a la generación y evaluación parcial de los clasificadores bayesianos.

Por otra parte, existen aplicaciones especializadas solo en la creación gráfica y

[1] *Splice-sites:* En los organismos eucariotas el ADN contiene en los genes segmentos codificantes (*exones*) y no codificantes (*intrones*). Se denomina splice-sites al borde de estos intrones.

manipulación de RB, ejemplo de ello es JavaBayes, herramienta libre desarrollada en Java, que permite editar visualmente una red, y realizar predicciones seleccionando uno de dos algoritmos de propagación de evidencia. La desventaja de este tipo de aplicaciones radica en que no permiten la generación del clasificador bayesiano aprendiendo de una base de conocimientos, ni consecuentemente su evaluación, así como imposibilita añadir los casos nuevos inferidos a una base de datos de conocimiento.

Teniendo en cuenta la panorámica planteada, estos algoritmos lamentablemente no son aplicados por la comunidad científica, puesto que no se dispone de una herramienta informática libre orientada al personal científico de áreas especializadas en bioinformática y biomedicina que **integre** en un ambiente unificado de trabajo la generación de clasificadores bayesianos utilizando los algoritmos de aprendizaje estructural existentes, y los nuevos algoritmos definidos para este tipo de dominio de estudio, de forma que posibilite al especialista evaluar el modelo aprendido, especificando las métricas a tener en cuenta para la comparación y selección del modelo que mejor se ajuste al problema, y su uso en la clasificación de nuevos casos, permitiendo la persistencia de los mismos, de tal manera que pueda usarla e interpretar los resultados desde su área del saber sin necesidad de conocer el procedimiento de fondo.

La ***problemática*** expuesta anteriormente se pude resumir en que no se dispone de una herramienta informática libre que integre en un ambiente unificado de trabajo, las tres funcionalidades que se relacionan a continuación, partiendo de la necesidad de las mismas acotadas a la investigación precedente llevada a cabo en la UCLV:

- Generación de clasificadores bayesianos a partir de una base de datos de entrenamiento, incluyendo nuevos algoritmos de aprendizaje estructural (ByNet, BayesChaid, BayesPSO) para dominios bioinformáticos y médicos.

- Evaluación de clasificadores bayesianos a partir de la selección de los métodos (incluyendo la validación cruzada aprovechando las potencialidades de un entorno controlado de tareas distribuidas) y las métricas (incluyendo la *exactitud*

y el *coeficiente de correlación de Matthews* [9], así como la *prueba no paramétrica de Friedman* [7]) que permita tener una visión integral y sencilla al especialista para la comparación y selección del modelo que más se adecue al dominio de estudio.

- Uso de los modelos generados permitiendo la clasificación de nuevos casos y la inclusión de los mismos en el conjunto de estudio.

De acuerdo a la situación problémica descrita anteriormente, se identifica el siguiente problema de investigación:

*¿Cómo realizar en un ambiente unificado de trabajo los procesos de **generación**, **validación** y **uso** de clasificadores bayesianos para dominios bioinformáticos y médicos?*

Partiendo de la interrogante planteada como problema de investigación, se identifica como **objeto de estudio** los *procesos de generación, evaluación y uso de los clasificadores bayesianos,* y dentro de este se selecciona como **campo de acción** los *procesos automatizados de generación, evaluación y uso de los clasificadores bayesianos en las aplicaciones informáticas para la clasificación de nuevos casos en dominios bioinformáticos y médicos.*

Para dar solución al problema de la investigación, se propone como **objetivo general***: Desarrollar una aplicación informática basada en tecnología libre, que **integre** en un ambiente unificado de trabajo los procesos de **generación** de clasificadores bayesianos, incluyendo los algoritmos ByNet, BayesChaid, BayesPSO, la **evaluación** de los mismos y su **uso** en la clasificación de nuevos casos.*

Del mismo se derivan los siguientes **objetivos específicos:**

- *Realizar el diseño e implementación de una aplicación informática que integre los procesos de generación, validación y uso de clasificadores bayesianos.*
- *Validar la aplicación informática propuesta mediante pruebas unitarias, pruebas de integración y pruebas de usabilidad.*

La investigación se desarrolla teniendo como **hipótesis**:

Desarrollar una aplicación informática que integre los procesos de generación de clasificadores bayesianos, incluyendo los algoritmos para dominios bioinformáticos y médicos, la evaluación, y su uso en la clasificación de nuevos casos, disminuirá la dificultad del uso de este tipo de modelo estadístico por parte de los especialistas.

Entre los **métodos de la investigación científica** utilizados se encuentran:

Métodos teóricos. El método analítico-sintético que se aplicó en los procesos listados a continuación:

- Estudio de la bibliografía relacionada e interpretación de los conceptos teóricos y prácticos afines a los clasificadores bayesianos.
- Descomposición del problema de investigación en el estudio independiente y generación de requisitos funcionales referidos a los procesos de generación, evaluación y uso de los clasificadores bayesianos, permitiendo a través de la síntesis, establecer las relaciones esenciales que vinculan a estos procesos.
- Análisis, interpretación, reutilización, y extensión de las clases y componentes brindados por Weka, para sintetizarlos en la solución general del problema planteado.
- Arribar a conclusiones de acuerdo a la investigación.

Métodos empíricos. El método de la medición, para obtener información numérica en el desarrollo de las pruebas de usabilidad de la solución, y el método experimental, para realizar las pruebas unitarias y de integración de la solución, mediante el desarrollo de casos de prueba experimentales previamente diseñados.

Como **técnicas de recopilación de información** se utilizó la entrevista con el propósito de obtener información a través de cuestionarios verbales sobre el tema de investigación y la encuesta para la realización de la validación de la solución propuesta mediante las pruebas de usabilidad.

Novedad y aporte teórico de la investigación:

- Modelo para facilitar el uso y aplicación de clasificadores bayesianos como representación estadística que se ajuste al dominio de estudio en investigaciones científicas relacionadas con la bioinformática y la medicina.
- Extensión de los algoritmos ByNet y BayesChaid con la implementación de otros procedimientos de cálculo del estadígrafo Chi cuadrado.

Novedad y aporte práctico de la investigación:

- La implementación de una aplicación informática que integre la generación de clasificadores bayesianos, incluyendo los algoritmos de aprendizaje estructural *BayesChaid, ByNet, BayesPSO,* la evaluación de los mismos a través de la selección y verificación por parte del usuario de los métodos y métricas definidas, y su uso en la clasificación de nuevos casos.
- La distribución de la validación cruzada sobre la plataforma de cálculo distribuido T-Arenal.
- La inclusión de funcionalidades que permiten la inserción de nuevos casos, clasificarlos, y que sus resultados tengan persistencia y puedan ser añadidos al conjunto inicial de entrenamiento para posteriores aprendizajes.
- Constituye una guía para la creación de software desde el punto de vista de la reutilización e integración de código, expresándolo con la extensión e integración de la biblioteca de clases Weka y su uso en la implementación de un software a la medida.

El presente documento se encuentra estructurado en tres capítulos que se describen a continuación:

Capítulo 1. Fundamentación teórica.
Son expuestos y referenciados los principales conceptos matemáticos que definen un clasificador bayesiano y cómo se efectúan los procesos de generación, evaluación e inferencia. Además se realiza un estudio de las principales plataformas de software que dan soporte a los mismos, así como se abordan los diferentes conceptos relacionados con la integración de aplicaciones.

Capítulo 2. Desarrollo del sistema.

En este capítulo se listan los requisitos funcionales y se realiza una comparación entre estos y las funcionalidades expuestas por Weka. Además se especifica la organización estructural del sistema a través de la arquitectura del mismo y la presentación del diseño e implementación de componentes específicos.

Capítulo 3. Validación de la solución.

En el presente capítulo se realiza la validación del sistema mediante pruebas unitarias, pruebas de integración y pruebas de usabilidad. Además se expone un breve manual de usuario del sistema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

La generación, evaluación y uso de los clasificadores bayesianos son procesos con un fuerte basamento teórico-matemático que es imprescindible entender para poder realizar la integración de estos procesos. En el presente capítulo son expuestos y referenciados los principales conceptos matemáticos que definen un clasificador bayesiano y cómo se efectúan los procesos de generación del mismo a través del aprendizaje estructural y paramétrico. Además se listan las diferentes métricas y métodos de evaluación, y se explica la realización de la inferencia en una red bayesiana.

Para integrar este tipo de procesos desde el punto de vista computacional se realiza un estudio de las principales plataformas de software que dan soporte a los mismos, así como se abordan los diferentes conceptos relacionados con la integración de aplicaciones.

1. Técnicas de clasificación de datos

La clasificación de datos es una operación estadística que permite el agrupamiento de las unidades de observación en *clases* (categorías, intervalos numéricos, modalidades). Se basa en la comprensión del dominio de aplicación y el establecimiento de patrones expresados en las relaciones existentes entre los atributos que definen las entidades, permitiendo sintetizar características comunes para definir por ejemplo funcionalidad y aplicación de las entidades que pertenecen a una misma clase.

Esta técnica ha sido aplicada en todas las ramas de la ciencia para la comprensión de las diferentes entidades que la componen y su planteamiento empírico data desde aproximadamente el año 350 a.C. donde se registra que Aristóteles realiza la primera clasificación de plantas y animales. Actualmente su valor práctico se evidencia en un amplio rango de aplicaciones, entre las que se encuentran: diagnóstico médico y psicológico, aplicaciones en economía, supervisión y diagnóstico de fallas en sistemas automáticos complejos, entre otros.

El problema de clasificación se basa en agrupar y discriminar objetos, descritos mediante un conjunto de atributos, ya sea construyendo las clases o asignando los objetos a clases previamente definidas. Para el mismo, se establece que las clases deben ser mutuamente excluyentes, o sea, una unidad de observación no puede ser clasificada simultáneamente en dos clases, con la misma escala, y a su vez, el conjunto de clases debe ser exhaustivo, no pudiendo ninguna unidad de observación quedar sin tener una clase donde pueda ser clasificada.

Comúnmente este tipo de estudios se aplica en análisis multivariado, donde surgen con mucha frecuencia exploraciones en las cuales se quiere conocer el efecto de un conjunto de variables discretas denominadas *variables independientes o predictoras*, sobre una variable que identifica la pertenencia a un grupo y que es supuestamente dependiente de ellas, denominada *variable clase o dependiente*, con el objetivo de obtener una clasificación. A pesar de que dicha variable dependiente o clase sea el blanco del estudio, las inferencias pueden ser no solo sobre la misma, sino sobre cualquiera de las *variables independientes* cuya información se desconozca a partir de evidencias de otras variables.

En términos matemáticos, el problema de clasificación, se podría plantear como la necesidad de asignar un vector de *p* observaciones $x = (x_1,x_2,...,x_p)$ a una de m clases $C_1,...,C_m$. En este problema se suponen conocidas las llamadas probabilidades a *priori* de las clases $P(C_1) ,... , P(C_m)$. Tales probabilidades indican la proporción de elementos existentes de cada clase. Por ejemplo, si el problema es de diagnosis médica y el 80% de las personas están sanas, y el 20% están enfermas (digamos de gripe), pues $P(C_1) = 0.8$ y $P(C_2) = 0.2$, donde C_1 = personas sin gripe , y C_2 = personas con gripe. En este caso, $x = (x_1,x_2,...,x_p)$ sería un vector de medidas tales como la temperatura corporal y otras.

Dado el problema de clasificación, se definen los clasificadores, modelos matemáticos que describen los datos y deben ser capaces de calcular las denominadas probabilidades a *posteriori*:

$P(C_1|x)$ = Probabilidad de que el individuo descrito por el vector *x* pertenezca a la clase C_1.

$P(C_2|x)$ = Probabilidad de que el individuo descrito por el vector *x* pertenezca a la clase C_2

.....

$P(C_m|x)$ = Probabilidad de que el individuo descrito por el vector *x* pertenezca a la clase C_m.

De este modo se asigna al individuo descrito por el vector *x* la clase que dé la mayor probabilidad a posteriori, llevando a cabo lo que se denomina predicción o inferencia.

En el problema de la gripe, si por ejemplo $P(C_1|x) = 0.9$ y $P(C_2|x) = 0.1$, entonces se concluye que el individuo pertenece al grupo de los sanos.

Por lo anteriormente expuesto, es que las técnicas de clasificación se utilizan comúnmente en situaciones donde están presentes:

1. Una población de datos que se dividen en dos o más clases de acuerdo con una taxonomía determinada.
2. Un grupo de datos de los cuales se conoce a *priori* la clase a la que pertenecen.
3. Un conjunto de datos de los cuales se desea saber a qué clase pertenecen.

El desarrollo tecnológico devenido en los últimos años ha potenciado a las investigaciones científicas con mecanismos automatizados de recolección y almacenamiento de datos, permitiendo que este aspecto pase a un segundo plano. Hoy el problema consiste en cómo obtener información a partir de estas fuentes masivas de datos caracterizadas por un alto porcentaje de redundancia y de datos ruidosos o sujetos a errores, insostenibles de analizar desde el punto de vista humano. Esto ha generado un avance en el diseño de técnicas de análisis automatizadas que aprovechan las capacidades de cómputo del ordenador en pos de obtener patrones o modelos a partir de los datos recopilados. Es así como surgen las principales técnicas de clasificación dentro de los algoritmos de aprendizaje automático que permiten generar clasificadores para su uso posterior.

La **generación** de un clasificador consiste en construir criterios para determinar el valor del atributo *clase* en un ejemplo cualquiera del dominio a partir de un conjunto de ejemplos, denominado *conjunto de entrenamiento*, de un cierto dominio *D*, aplicando

un algoritmo de aprendizaje,. Esos criterios están basados en los valores de uno o varios de los pares (atributo; valor) que intervienen en la definición de los ejemplos. El **uso** de un modelo clasificador estará determinado por su capacidad para predecir o inferir la *clase* a la que pertenece una nueva entidad correspondiente al dominio de estudio (ver Fig.1).

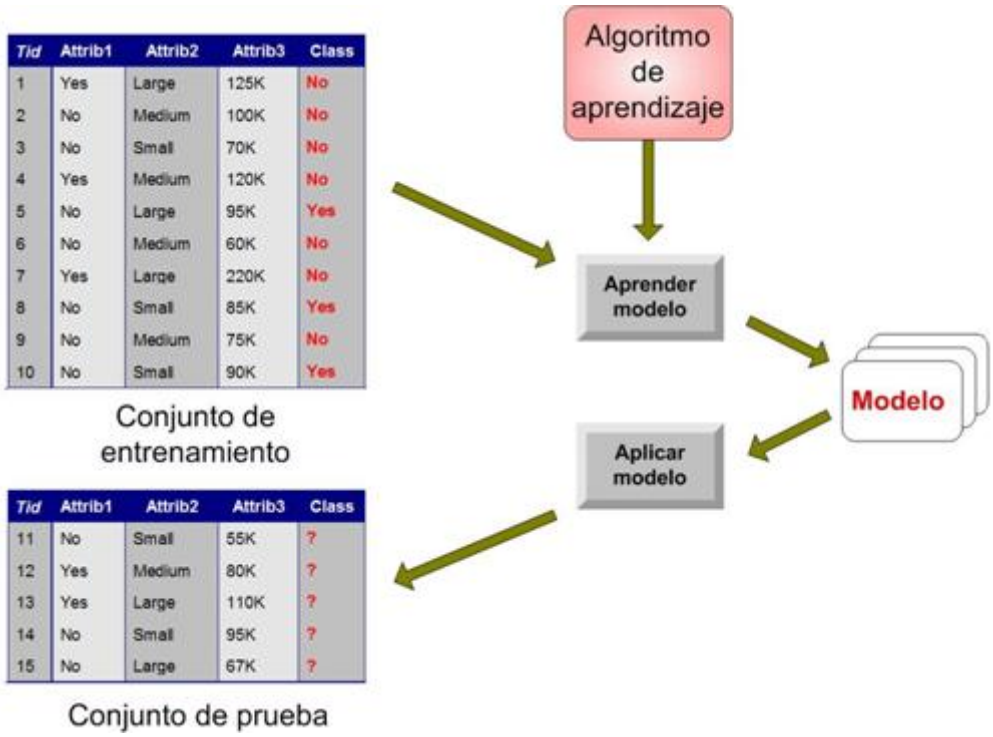


Fig.1 Proceso de generación y uso de un clasificador.

Entre las principales técnicas automatizadas de clasificación se encuentran los clasificadores bayesianos.

2. Clasificadores bayesianos

Los clasificadores bayesianos [10] son clasificadores estadísticos, que pueden predecir la probabilidad de que una muestra dada pertenezca a una clase particular. Este tipo de clasificador utiliza como modelo matemático una Red Bayesiana conjuntamente con el Teorema de Bayes [11].

2.1. Redes bayesianas

Las RB (también conocidas como redes causales probabilísticas, redes causales, sistemas expertos bayesianos, redes de creencia, sistemas expertos probabilísticas o diagramas de influencia) son una representación gráfica de dependencias para razonamiento probabilístico, que combina la potencia del Teorema de Bayes con la expresividad semántica de los grafos dirigidos permitiendo expresar un modelo causal de las independencias/dependencias entre las variables que forman parte del dominio de aplicación [12]. De forma resumida se puede decir que una RB es un conjunto de variables, una estructura gráfica conectando estas variables y un conjunto de distribuciones de probabilidad condicional. Codifica incertidumbre asociada a cada variable por medio de probabilidades y, gracias al Teorema de Bayes [11], esta incertidumbre es susceptible de ser modificada con base en observaciones (o evidencias) sobre el modelo.

Formalmente se define un modelo de Red Bayesiana, como un par (*G, P*), donde *G* es un grafo acíclico dirigido (GDA)*, y P* una distribución de probabilidad conjunta (*DPC). P*= {$p(X_1|\tau_1)$*,* $p(X_2|\tau_2)$*, …,* $p(X_n|\tau_n)$} es un conjunto de *n* distribuciones de probabilidad condicionales, una por cada variable X_i(nodos del grafo), y τ_i es el conjunto de padres del nodo X_i en *G*. [13]

Partiendo del enunciado de distribución de probabilidad conjunta para las variables, si tenemos una red con N nodos y con variables binarias, haría falta conocer 2^N-1 valores. Sin embargo, las condiciones de independencia dadas por la separación direccional (Hipótesis de Independencia Condicional o de Separación Direccional [14]) permiten que no sea necesario conocer todos estos valores, y aplicando el Teorema Factorización de la Probabilidad [14], la distribución de probabilidad conjunta se puede expresar como producto de las distribuciones condicionadas de cada nodo dados sus padres. [14] Aplicando lo anterior, el conjunto *P* define la DPC asociada, como muestra la expresión:

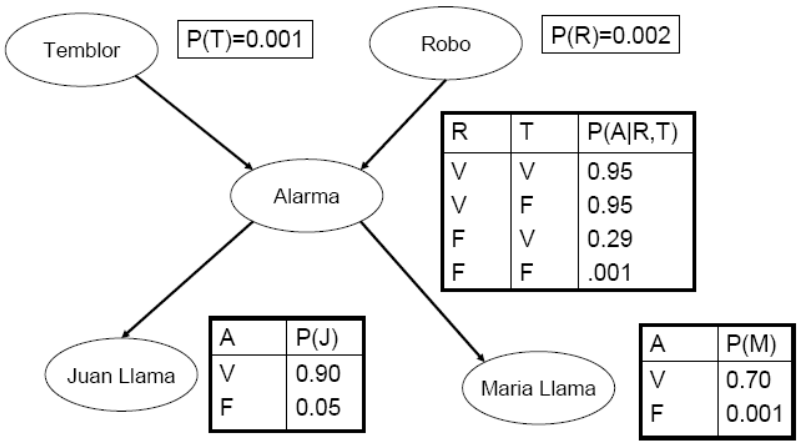
$$p(X) = \prod_{i=1}^{n} p(X_i|\boldsymbol{\tau}_i) \qquad X = (X_1, X_2, ..., X_n)$$

Una RB incluye como componentes básicos (ver Fig.2).

• Un conjunto de nodos $\{X_i\}$ que representan cada una de las variables del modelo. Cada una de ellas tiene un conjunto exhaustivo de estados $\{x_i\}$ mutuamente excluyentes. Estas variables pueden representar rasgos o atributos.

• Un conjunto de enlaces o arcos dirigidos $\{X_i, X_j\}$ entre aquellos nodos que tienen una relación causal. El significado de un enlace que va del nodo *X* al nodo *Y* es el de que *X* ejerce una influencia directa sobre *Y*. En términos de probabilidades esto significa que hay una dependencia condicional de *Y* respecto a *X*, o sea que la probabilidad de *Y* es diferente de la probabilidad de *Y* dado *X*. De esta manera todas las relaciones están explícitamente representadas en el grafo.

• Una tabla de probabilidad condicional asociada a cada nodo X_i indicando la probabilidad de sus estados (parámetros) para cada combinación de los estados de sus padres. Si un nodo no tiene padres se indican sus probabilidades a *priori*.



R	T	P(A|R,T)
V	V	0.95
V	F	0.95
F	V	0.29
F	F	.001

A	P(J)
V	0.90
F	0.05

A	P(M)
V	0.70
F	0.001

Fig.2 Red bayesiana generada para el estudio de ejecución de alarmas.

2.2. Aprendizaje de una red bayesiana

La definición de una RB está determinada por dos tareas. La primera, denominada aprendizaje estructural, consiste en obtener la estructura de grafo a partir de las relaciones de dependencia condicional entre las variables. La segunda tarea, caracterizada por calcular la distribución de probabilidades (parámetros) que permitirá hacer inferencias, se denomina aprendizaje paramétrico.

2.2.1. Aprendizaje estructural

Hay distintos enfoques para obtener la estructura de la red bayesiana, bien porque la red se conoce de antemano o bien porque se infiere de los datos de entrenamiento, y por otro lado si todas las variables que intervienen en la red son observables o bien si hay algunas que no lo son.

La forma básica para la generación de la estructura de bayesiana se adjudica al clasificador bayesiano simple Naive Bayes (NB), el cual asume que los atributos son condicionalmente independientes entre sí dado el atributo clase, de tal manera que no existen arcos entre ellos, y la probabilidad se puede obtener por el producto de las probabilidades condicionales individuales de cada atributo dado el nodo clase.

El clasificador bayesiano simple asume que los atributos son independientes dada la clase. Si esto no es verdad, una forma de considerar estas dependencias es extendiendo la estructura básica de NB agregando arcos entre dichos atributos (algoritmos TAN [15] y BAN [15]).

Las técnicas de aprendizaje estructural son diversas y dependen del tipo de estructura de red: árboles [16], poli-árboles [17] y redes múltiplemente conexas (métodos basados en medidas de ajuste y búsqueda: K2 [18] y el algoritmo B [19], y los métodos basados en pruebas de independencia: el algoritmo Power Constructor [20].

Otra alternativa es combinar conocimiento subjetivo del experto con aprendizaje. Para ello se parte de la estructura dada por el experto, la cual se valida y mejora utilizando datos estadísticos. Resulta evidente que la calidad de una red obtenida de esta manera depende mucho del conocimiento sobre el dominio de aplicación de los encuestados. Esta última opción no siempre es aplicable en problemas bioinformáticos.

En general ninguno de los algoritmos para obtener la estructura de una RB es considerado mejor que otro, depende del dominio específico de estudio y el problema que se quiere resolver. En [8] se realiza un análisis de los algoritmos y su inadecuación con respecto al dominio de investigaciones bioinformáticas y biomédicas, por lo que proponen y validan el uso de tres nuevos algoritmos de aprendizaje estructural: ByNet, BayesChaid, y BayesPSO.

El aprendizaje estructural dentro de las tareas que se llevan a cabo para definir una RB, es esencial por ser realmente la más difícil e imprescindible para realizar el aprendizaje paramétrico. Si a lo anterior se añade que sin tener en cuenta el método seleccionado para generar la estructura de red bayesiana, el número de modelos (redes bayesianas) diferentes que son posibles, se eleva de manera considerable en función del número de variables involucradas en una razón de:

$$2^{\frac{n * (n-1)}{2}}$$

donde *n* representa el número de variables, se tiene una evidencia más para afirmar que las posibilidades del uso de una RB se fortalecen especialmente si se logra optimizar el aprendizaje estructural acorde con el dominio del campo de aplicación [8].

2.2.2. Aprendizaje paramétrico

Una vez conocida la estructura del grafo, se pasa a la tarea de obtener las probabilidades (parámetros) correspondientes a cada nodo de la red bayesiana, las probabilidades *a priori* de los nodos raíz y las probabilidades condicionales de las demás variables, dados sus padres.

Cuando se tienen datos completos y suficientes para todas las variables en el modelo, es relativamente fácil obtener los parámetros, asumiendo que la estructura está dada. El método más común es el llamado *estimador de máxima verosimilitud*, bajo el cual se estiman las probabilidades sobre la base de las frecuencias de los datos de entrenamiento. Para una red bayesiana se tienen dos casos:

- Nodos raíz. Se estima la probabilidad marginal. Por ejemplo: $P(A_i) \sim NA_i/N$, donde NA_i es el número de ocurrencias del valor *i* de la variable *A,* y *N* es el número total de casos o registros.

- Nodos hoja. Se estima la probabilidad condicional de la variable dados sus padres. Por ejemplo: $P(B_i \mid A_j, C_k) \sim NB_iA_jC_k / NA_jC_k$, donde $NB_iA_jC_k$ es el número de casos en que $B = B_i$, $A = A_j$ y $C = C_k$, y NA_jC_k es el número de casos en que $A = A_j$ y $C = C_k$.

La calidad de estas estimaciones dependerá de que exista un número suficiente de datos en la muestra. Cuando esto no es posible se puede cuantificar la incertidumbre existente representándola mediante una distribución de probabilidad (usualmente la distribución Beta [21] y/o Dirichlet [22])

Otro aspecto importante es el proceso de discretización. Normalmente las RB utilizan variables discretas o nominales, por lo que si no lo son, hay que discretizarlas antes de construir el modelo estructural.

2.3. Inferencia en una red bayesiana

La inferencia se refiere a obtener conclusiones basadas en premisas o evidencias, permitiendo realizar predicciones en base a las nuevas probabilidades.

La propagación de evidencias en la red, se realiza si se tiene en cuenta un conjunto de variables *evidenciales* $E \subseteq D$ con valor evidencial $E = e$ (*e representa uno de los posibles valores de la variable de evidencia*) y el conjunto de variables no evidenciales $D \mid E$ para las cuales se calculan las probabilidades condicionales $P(X_i \mid e)$.Una forma de calcular $P(Xi \mid e)$ en una RB, es aplicando el Teorema de Bayes, herramienta que permite expresar la probabilidad condicional de un evento e ir actualizando las probabilidades en base a los valores establecidos a ciertas variables.

Para describir el proceso de inferencia, se debe tener en cuenta que en primer lugar hay que compilar la red para que se cree una representación interna de las probabilidades del modelo (aprendizaje paramétrico). De esta manera se estima lo que se conoce como distribución previa (o distribución *a priori*) del modelo. A continuación se van añadiendo evidencias sobre el estado de las variables del modelo y se van obteniendo sucesivamente las distribuciones *a posteriori*. El proceso de clasificación se realiza entonces asignando a las variables actualizadas la observación en aquella clase que tenga la mayor probabilidad.

Si se intenta afrontar todos los cálculos requeridos para la inferencia en una topología general de RB aplicando el Teorema de Bayes [11], sería necesario realizar un número de operaciones que crece exponencialmente con el número de variables de la red, y se

convertiría en una tarea computacionalmente ineficiente, por lo que se han diseñado algoritmos de propagación que tienen en cuenta las estructuras de independencias entre las variables, reduciendo el número de cálculos considerablemente.

Los diferentes algoritmos de propagación se agrupan teniendo en cuenta el tipo de grafo y si se obtiene la probabilidad de una variable a la vez o la de todas. Dentro del grupo que puede predecir cualquier variable, partiendo de cualquier tipo de estructura general, se encuentran los algoritmos de agrupamiento, siendo uno de los algoritmos más representativos dentro de este grupo el de *árboles de unión o junction tree* [12] en su definición en inglés.

El algoritmo de árboles de unión se basa en el envío de *mensajes* en un árbol de unión construido a partir de la transformación de la estructura de la red para obtener un árbol, mediante la construcción de subconjuntos de nodos (llamados conglomerados o cliques) usando la teoría de grafos. De esta forma, el proceso de propagación de evidencia puede ser acometido calculando probabilidades locales (que dependen de un número reducido de variables), evitando así calcular probabilidades globales (que dependen de todas las variables).

2.4. Ventajas de los clasificadores bayesianos

Son diversas las ventajas referidas a las RB como modelo matemático para la representación de diferentes dominios de estudio [23] [15] [24]. Entre ellas se puede mencionar que las RB proveen de una representación gráfica de las relaciones explícitas de dependencia del dominio, de manera que permiten descubrir la estructura causal subyacente en un conjunto de datos [25] a partir de una base de datos que contenga un conjunto de observaciones sobre un conjunto de variables, realizando el modelado de sistemas complejos y visualizando las relaciones causales por medio del grafo.

Además las RB permiten la inferencia bidireccional, con lo que se logra realizar inferencias en ambos sentidos, lo que significa que estas pueden ser no solo sobre la “clase o variable dependiente” sino sobre cualquiera de las variables “independientes” cuya información se desconozca a partir de evidencias de otras variables [24]. Esto se

puede realizar conjuntamente con el uso de valores con grados de certidumbre, por lo que se pueden generar buenas predicciones con información o evidencias incompletas.

Dada la incertidumbre presente en los datos provenientes de investigaciones bioinformáticas y biomédicas, como en los estudios de análisis de secuencias genómicas u otros datos biológicos (p.e. problema de clasificación de la resistencia de una nueva mutación y diagnóstico probabilístico diferenciado de una determinada enfermedad) [8], y analizando las ventajas que ofrece el uso de las RB sobre las técnicas estadísticas y bioinformáticas convencionales [24], resulta apropiado el uso de clasificadores bayesianos.

Partiendo del análisis anterior y las bondades que presentan las RB, surge la idea por parte del Grupo de Bioinformática de la UCLV de trabajar con este tipo de técnica, desarrollando nuevos algoritmos de aprendizaje estructural: ByNet, BayesChaid, y BayesPSO, con el objetivo de simplificar la estructura de la red, tomando como apoyo otros modelos gráficos probabilísticos o de optimización, así como teniendo en cuenta características concretas del dominio de aplicación para aliviar el cálculo de probabilidades, facilitar inferencias y reducir complejidad computacional, siendo particularmente aplicables en estudios bioinformáticos y biomédicos y con eficiencia similar o superior a los algoritmos ya existentes [8].

Dos de estos algoritmos (ByNet y BayesChaid) obtienen la estructura de dependencias basándose en la detección de interacciones al estilo del algoritmo CHAID (Chi-square Automatic Interaction Detector). El tercero, BayesPSO, se basa en un método de optimización bioinspirado, concretamente en el de optimización basado en enjambres de partículas (Particle Swarm Optimization, PSO) para contribuir a la reducción de atributos.

3. Evaluación de los clasificadores

Uno de los aspectos que se tienen en cuenta a la hora de evaluar un clasificador bayesiano es su precisión, el cual se encuentra relacionado con el porcentaje de casos clasificados correctamente. Para ello es necesario definir ciertos criterios de comparación, y los resultados obtenidos dependerán en gran medida de dos aspectos,

el método de clasificación seleccionado y el conjunto de datos disponible para entrenar el clasificador, teniendo en cuenta el grado de representatividad que tengan estos últimos sobre los datos reales.

Antes de construir el modelo de clasificación, el conjunto de datos disponible se divide en un conjunto de entrenamiento (para construir el modelo) y un conjunto de prueba (para evaluar el modelo). Una vez construido el modelo, su evaluación se basa en el uso del mismo para clasificar los datos del conjunto de prueba, comparando los casos etiquetados del conjunto de prueba inicial con el resultado de aplicar el modelo. Es así como se obtiene un porcentaje de casos clasificados correctamente que cuantifica que tan aceptable es la precisión del clasificador para utilizarlo en la clasificación de nuevos casos.

En este contexto, aparecen varios procedimientos para seleccionar de los datos disponibles, los datos que son de entrenamiento y los que son de prueba, aspecto al que se refieren los métodos de evaluación. Por otra parte se encuentran las diferentes formas de representar cuantitativamente la precisión de un clasificador, las cuales se denominan métricas de evaluación.

3.1. Métricas de evaluación

De forma general las métricas de evaluación están basadas en los cálculos generados a partir de la matriz de confusión.

La matriz o tabla de confusión expone los resultados de un procedimiento de clasificación o predicción. Si hay k categorías posibles, es una matriz k x k donde las filas denotan la verdadera categoría y las columnas indican la categoría a la que se asigna después de la predicción. El elemento *ij* define el número de muestras de la categoría *i* que fueron clasificados como categoría *j*, representando así los errores en la asignación de categorías a los patrones actuales. Aquellas entradas en la diagonal de la matriz son correctas. Las entradas fuera de la diagonal son errores de clasificación y representan “confusión”. Con la matriz es fácil ver si el sistema confunde dos categorías.

		Predicción		Error del modelo
		C_P	C_N	
Clase real	C_P	**TP**	**FN**	FN/(FN+TP)
	C_N	**FP**	**TN**	FP/(FP+TN)
Error de uso		FP/(FP+TP)	FN/(FN+TN)	Error General (FP+FN)/ (TN+TP+FP+FN)

Fig.3 Matriz de confusión extendida con la representación de algunos tipos de errores.

Las cuatro posibilidades que pueden resultar de una predicción cuando hay dos categorías "positivo" o "sí" y "negativo" o "no" son:

- Falsos positivos (FP, False positive): El resultado es incorrectamente clasificado como "sí" o "positivo" cuando en realidad es "no" o "negativo".
- Falsos negativos (FN, False negative): El resultado es incorrectamente clasificado como negativo cuando en realidad es positivo.
- Verdaderos positivos (TP, True positive): El resultado es correcto, representa la cantidad de entidades clasificadas correctamente como positivos.
- Verdaderos negativos (TN, True negative): El resultado es correcto, representa la cantidad de entidades clasificadas correctamente como negativas.

Las principales métricas definidas a partir de la matriz de confusión son las siguientes:

- Exactitud del clasificador (accuracy)

Constituye la proporción del número total de predicciones que son correctas. Se define como:

exactitud = (TP+TN)/(TP+TN+FP+FN)

Cabe señalar que la exactitud puede no ser una medida adecuada para evaluar el comportamiento cuando el número de casos de un tipo es mucho mayor que el número de casos contrario, ya que este índice es global y no indica cómo la exactitud se reparte entre las diversas categorías individuales.

- Precisión (precision)

Se corresponde con la proporción de casos positivos predichos que son correctos. Se determina utilizando la ecuación:

precisión =TP/(TP+FP)

- Razón de verdaderos positivos (rTP: True positive recognition rate)

La tasa de verdaderos positivos, también llamada sensibilidad, es la proporción de casos positivos que están correctamente identificados. Se define mediante la siguiente ecuación:

rTP = TP/P = TP/(TP+FN)

- Razón de verdaderos negativos (rTN: True negative recognition rate)

La tasa de verdaderos negativos (rTN), también denominada *especificidad*, se define como la proporción de casos negativos que fueron clasificados correctamente. Puede decirse que constituye un método para evaluar la exactitud de las categorías individuales, específicamente la categoría positiva. Se determina como:

rTN= TN/N = TN/(TN+FP)

- Razón de falsos positivos (rFP: False positive recognition rate)

La tasa de falsos positivos (rFP) es la proporción de casos negativos que fueron incorrectamente clasificados como positivos. Al igual que el rTP, puede decirse que constituye un método para evaluar la exactitud de las categorías individuales, en este caso la categoría negativa. Se determina utilizando la ecuación:

rFP = FP/(FP+TN)

- Razón de falsos negativos (rFN: False negative recognition rate)

La tasa de falsos negativos (rFN) es la proporción de casos positivos que fueron erróneamente clasificados como negativos. Se determina utilizando la ecuación:

rFN = FN/(FN+TP)

- Coeficiente de correlación de Matthews (MCC: Matthews Correlation Coefficient)

El Coeficiente de correlación de *Matthews* es una medida de la calidad utilizada en

máquinas de aprendizaje para clasificaciones, que retorna un valor entre -1 y +1. Un valor de +1 representa una predicción perfecta, 0 representa una predicción aleatoria y -1 promedia una predicción inversa. El MCC generalmente se considera una medida equilibrada que se puede utilizar incluso si las clases son de tamaños muy diferentes. Su expresión queda definida de la siguiente forma:

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}$$

- Área bajo la curva ROC (ROC Area: Area Under ROC Curve)

Otra forma de evaluar el rendimiento de un clasificador es utilizando el área bajo las curvas ROC (*Receiver Operator Characteristic)* [26], donde se realiza una representación gráfica de la razón de verdaderos positivos (rTP), frente a la razón de falsos positivos (rFP), como ejes X e Y respectivamente. Dado que la rTP es equivalente a *sensibilidad* y la rFP lo es a (*1-especificidad*), el gráfico ROC se define también como la representación de (*1-especificidad*) frente a la *sensibilidad*.

3.2. Métodos de evaluación

Existen varias formas de dividir los datos disponibles en datos de entrenamiento y datos de prueba:

- Usar la propia base de datos de entrenamiento para pruebas.
- Utilizar una base de datos independiente para pruebas.
- Utilizar un subconjunto o porcentaje del conjunto de datos disponibles para entrenamiento y otro para prueba de forma estática.
- Utilizar un subconjunto del conjunto de datos disponibles para entrenamiento y otro para prueba de forma alternada e iterativa (*validación cruzada*).

Cabe señalar que la primera opción no es altamente fiable, ya que aunque el conjunto de entrenamiento no se corresponda con una adecuada muestra representativa de los datos reales, la evaluación puede generar resultados alentadores que no son ciertos a la hora de aplicar el clasificador a nuevos casos no presentes en el conjunto de entrenamiento.

La validación cruzada es uno de los métodos de evaluación más abarcadores y de más amplio uso. La misma consiste en dividir aleatoriamente el conjunto de datos en *k* subconjuntos de intersección vacía (más o menos del mismo tamaño). En la iteración *i*, se usa el subconjunto *i* como conjunto de prueba y los *k-1* restantes como conjunto de entrenamiento, de forma que la evaluación se realiza iterativamente y como medida de evaluación se toma la media aritmética de las *k* iteraciones realizadas. Existen diferentes variantes para la validación cruzada, entre ellas *k-fold Cross-Validation, LeaveOneOut*, y *Stratified Cross-Validation* [27] .

3.3. Comparaciones

Para comparar el rendimiento relativo de dos o más modelos de clasificación alternativos sobre un mismo conjunto de datos, basta la observación del comportamiento de algunas de las métricas señaladas como resultado del proceso de evaluación, siendo suficiente la presentación tabular de los resultados o su representación gráfica.

Cuando se dispone de varios conjuntos de prueba, o varios conjuntos de datos sobre los que se quiere evaluar el rendimiento relativo de estos modelos, entonces el empleo de pruebas estadísticas permiten ampliar el alcance de la comparación. En este caso se recomienda la realización de pruebas no paramétricas, siendo una de las más utilizadas la prueba de Wilcoxon (para la comparación de dos clasificadores) [7] y la prueba de Friedman (para comparaciones globales) [28].

Prueba de Friedman

La prueba de Friedman [7] constituye una prueba no paramétrica de propósito general que permite realizar análisis de diseños en bloques al azar.

Esta se puede aplicar para analizar si hay diferencias significativas entre los resultados obtenidos por diferentes clasificadores para diferentes conjuntos de entrenamiento. Básicamente consiste en que a partir de *k* variables en columnas (clasificadores) y *n* elementos en filas (bases de conocimiento con que se entrenaron los clasificadores), siendo el valor de la intersección el valor correspondiente a una de las métricas de evaluación, son ordenadas las fila de menos a mayor según las diferentes columnas

desde 1 hasta *k* (esto será el rango que ocupa cada variable para ese caso). Si no hubiera diferencias entre las variables se espera que los rangos estén repartidos en cada columna de manera uniforme y solo se encontrarán pequeñas diferencias entre ellas debidas al azar. La hipótesis nula de la prueba consiste en que los rangos sumados para cada columna (cada variable) son iguales, mientras que la alternativa sería que al menos uno de los rangos sumados es diferente.

4. Aplicaciones informáticas de generación y uso de clasificadores bayesianos

A partir del auge de la teoría de Redes Bayesianas y su aplicación, han sido múltiples los productos de software que se han desarrollado para su uso. Algunos de los pioneros fueron el resultado de grandes y costosos proyectos de investigación como por ejemplo: *Netica*, *Elvira*, *Hugin*.

4.1. Netica

Netica [3] es un programa para trabajar con redes bayesianas y diagramas de influencia. Esta soportado por Norsys, empresa Canadiense que se especializa en software de redes bayesianas. Cuenta con una interfaz de usuario intuitiva para la elaboración y edición de las redes, y las relaciones entre las variables se pueden entrar como probabilidades individuales, en forma de ecuaciones, o extraídas de archivos de datos. Una vez que se crea una red, Netica puede utilizar el algoritmo de árboles de unión para realizar inferencia. Además puede probar el rendimiento de una red mediante un archivo de los casos entrado, generando parámetros de salida como la matriz de confusión.

Netica-J: versión de la API (Application Program Interface) de Netica para Java, es una interface que permite reutilizar toda la potencia de Netica. Presenta a además una completa documentación textual y en el código java, y una guía de instalación. Da soporte la escucha de eventos para cualquier tipo de evento lanzado como la creación, eliminación, duplicación, etc. de las redes o los nodos, y soporte para operaciones de flujo de entrada y salida.

Lamentablemente tanto Netica como Netica Java-API son plataformas comerciales, y aunque pueden ser descargadas en una versión trial libre desde el sitio oficial están

limitadas en sus funcionalidades y en el tamaño del modelo que puede generar.

4.2. HUGIN

El programa comercial HUGIN [28], [2], desarrollado por investigadores vinculados a la Universidad de Aalborg y financiado por el consorcio HUGIN EXPERTS A/S, es considerado como una de las herramientas más eficaces para el desarrollo y la computación de redes bayesianas. Este software presenta una amplia gama de paquetes entre los que se encuentran Hugin Explorer, que brinda dos aplicaciones: HUGIN Graphical User Interface 7.4 - HGUI 7.4 y HUGIN Decision Engine 7.4 - HDE 7.4. Entre sus características se encuentran:

- Permite estimar la estructura, utilizando algunos algoritmos como PC, el de Chow-Liu y TAN, y la distribución de probabilidad condicional de una red bayesiana a partir de datos de entrada.
- Acepta la especificación de conocimiento del dominio de expertos (es decir, las limitaciones estructurales, incluido el apoyo para agregar restricciones entre un nodo y un subconjunto de nodos).
- Aplicación del algoritmo de árboles de unión de acuerdo al dominio, permitiendo especificar parámetros como tamaño y peso de los conglomerados, entre otros.
- Permite la edición de la red bayesiana y las tablas de probabilidad condicional.

Hugin presenta una versión limitada libre de demostración llamada Hugin Lite. Esta versión incluye ambas facilidades, HUGIN Graphical User Interface y Hugin Decision Engine, y está disponible con interfaces de programación de cinco ambientes diferentes: la API de C, C++ API, .NET API, la API de Java y ActiveX-Server para Visual Basic. Además contiene una biblioteca completa de bases de conocimiento pre-construidas a partir de diferentes áreas de negocio. Sin embargo, Hugin Lite tiene la limitante de solo estar preparada para manejar como máximo 50 estados y aprender como máximo de 500 casos, lo que resulta restringido para resolver problemas de bioinformática. Además de esta limitación, su uso está legalmente prohibido para cualquier otro propósito que no sea la demostración de sus capacidades.

4.3. Elvira

Elvira [1] [29] es fruto de un proyecto de coordinado de investigación desarrollado entre

los años 1997 y 2000, en el que participaron investigadores de varias universidades españolas y de otros centros, para la investigación de nuevos métodos y algoritmos de razonamiento probabilístico y la implementación de sistemas expertos bayesianos. El mismo está escrito en Java, utiliza un formato propio para la codificación de los modelos, y dispone de una interfaz gráfica para la construcción de redes. Además ofrece la posibilidad de hacer uso de un amplio abanico de algoritmos y permite realizar el aprendizaje de modelos a partir de bases de datos.

Sin embargo, para los usuarios, el problema principal de Elvira es la falta de robustez (pues aún presenta algunos fallos, principalmente en la interfaz gráfica), la falta de eficiencia, pues todavía necesita bastante trabajo de depuración, y la falta de documentación en línea. Para los programadores, la dificultad principal estriba en la falta de documentación (los comentarios del código del programa son muy útiles, pero insuficientes) y en que el código, por no haberse desarrollado con los principios de la ingeniería del software, resulta difícil de mantener.

4.4. Otras aplicaciones informáticas

Existen otras muchas herramientas informáticas que nos permiten trabajar con Redes Bayesianas y modelos probabilísticos basadas en las mismas. Estos han sido relacionados y comparados por diferentes autores siendo Murphy [30] uno de los más destacados en las referencias bibliográficas del tema, el cual hace una comparación detallada de 50 productos de *software* de RB. De los mismos se recoge 14 parámetros diferentes, dentro de los cuales son prioritarios para el análisis y desarrollo de esta investigación los que relacionan a continuación:

- Libre (parámetro Free): Dada las políticas de socialización del conocimiento y la imposibilidad de costear software propietario, la selección de una plataforma de desarrollo debe ser de código abierto y gratuito.
- Multiplataforma (parámetro Exec): Partiendo de las políticas anteriores, el software debe ser posible ejecutar sobre plataformas privativas como SO Windows, MAC, y libres como Linux.
- API: El software debe tener disponible una interface que permita interactuar y reutilizar el código del programa, para integrar las funcionalidades ya

implementadas en un código propio.

- Incluye el código fuente (parámetro Src): Tener el código fuente permite realizar una mejor reutilización e interoperabilidad, partiendo de la adaptación y comprensión del mismo.
- GUI: El software para su incrementar su facilidad de uso debe tener una interfaz gráfica de usuario (Graphical User Interface).

Desde el punto de vista funcional para dar respuesta a los requisitos de la investigación, se deben tener en cuenta además los siguientes parámetros:

- Aprendizaje Estructural (parámetro Learns structure?): El software debe ser capaz de realizar el aprendizaje estructural de la red bayesiana.
- Aprendizaje Paramétrico (parámetro Learns parameters?): El software debe ser capaz de realizar el aprendizaje paramétrico de la red bayesiana.
- Inferencia (parámetro Inference): La aplicación debe ser capaz de implementar la inferencia en la RB para su uso en la clasificación de nuevos casos.

Basándose los parámetros establecidos anteriormente y su prioridad, se realizó un primer filtro de la lista de software tomando en cuenta los parámetros del 1 al 3, resultando que 14 aplicaciones cumplían estos requisitos. Sobre el filtro anterior se consultó cuantas realizaban aprendizaje estructural resultando en dos aplicaciones, y cuantas realizaban aprendizaje paramétrico, devolviendo una lista de tres aplicaciones, siendo coincidentes con los dos parámetros solicitados solo una de ellas (ver Anexo 1), BNT, que además presenta varios métodos de inferencia, sin embargo el código fuente del mismo está desarrollado en C, lenguaje complejo de programación, primario dentro de los considerados de alto nivel, y Matlab, software propietario, y no presenta GUI.

4.5. ByShell

ByShell [8] [31] fue un software desarrollado bajo tecnología privativa, Borlan Delphi, por el equipo de la Universidad de las Villas con el objetivo de realizar inferencias sobre los modelos generados previamente con otras herramientas a partir del uso de los nuevos algoritmos desarrollados. Este presenta implementado el algoritmo de propagación de árboles de unión, pero no se encuentra totalmente validado.

4.6. JavaBayes

Dentro de las aplicaciones especializados en la creación gráfica y manipulación de Redes Bayesianas, mencionadas en el listado de Murphy [30] y descartadas en el filtrado de aplicaciones realizado en el acápite 4.3 se encuentra JavaBayes [6]. Esta herramienta, desarrollada completamente con tecnología libre, Java, y bajo la licencia GNU GPL. Trabaja con un modelo de RB previamente procesado o que se puede editar gráficamente, pero no permite la generación a partir del conocimiento presente en una base de datos, ni la evaluación de los modelos con relación a otros aplicando determinadas métricas. Sin embargo es una aplicación que permite realizar inferencias en la RB, y es recomendada por Weka [32] para este tipo de tarea.

Este sistema está compuesto por un editor gráfico que permite crear y modificar la RB y un núcleo de inferencia responsable de manipular las estructuras de datos, permitiendo calcular la probabilidad marginal y la esperada de cualquier variable en la RB usando dos tipos de algoritmos de propagación: eliminación de variables y el de árboles de unión.

Sin embargo, cuando se analizó JavaBayes v0.346 se encontraron ciertas deficiencias que lo hacen difícil de utilizar:

En el diseñador gráfico de la red para realizar las operaciones de creación, traslación, eliminación, edición de variables, y el cálculo de la probabilidad de alguno de los nodos, siempre se debe definir previamente presionando el botón correspondiente a la acción, y después es que se permite hacer la selección del nodo en cuestión, lo cual hace tedioso y poco interactiva la edición.

El diseño de la red y la salida de las operaciones se hacen en ventanas diferentes, que no tienen un foco visual común con respecto a las otras ventanas visualizadas en el escritorio del SO. Al ejecutar las operaciones de cálculo de probabilidades, el comando correspondiente se encuentra en la ventana secundaria, y después hay que pasar a la ventana de edición gráfica, definir que la función que se va a ejecutar es una *consulta* (*query*) y seleccionar con el mouse el nodo correspondiente, haciendo la operación complicada de ejecutar por parte del usuario.

La documentación del software plantea que permite abrir y salvar el modelo en formato XMLBIF versión 0.1, 0.2 y 0.3, sin embargo cuando se intenta abrir una red generada por Weka 3.6.2 definida por el formato XMLBIF 0.3, da un error y no es capaz de cargarla.

4.7. Weka

WEKA [32] [5], acrónimo de Waikato Environment for Knowledge Analysis, es una herramienta para el aprendizaje automático y minería de datos. Nace del esfuerzo de un grupo de investigadores del Machine Learning Laboratory de la Universidad de Waikato en Nueva Zelanda, como software de código abierto bajo los términos de la GNU GPL y constituye un conjunto de bibliotecas de código en Java para la extracción de conocimientos desde bases de datos, que incorpora disímiles técnicas estadísticas o de Inteligencia Artificial, y brinda la posibilidad de experimentar con el conjunto de ellas para investigar con cuáles se obtienen mejores resultados.

La extensa colección de algoritmos generales implementados en Weka, son útiles para ser aplicados mediante las interfaces que ofrece o para utilizarlo dentro de cualquier aplicación, ya que una de las propiedades más interesantes de este software, es su facilidad para añadir extensiones, modificar métodos etc. La misma contiene herramientas para realizar transformaciones sobre los datos, tareas de clasificación, regresión, agrupamiento, asociación y visualización. Todo lo anteriormente mencionado ha impulsado a que Weka sea una de las suites más utilizadas en el área en los últimos años.

Con relación a las funcionalidades relacionadas con algoritmos de aprendizaje de clasificadores bayesianos que tiene implementados, presenta las siguientes capacidades:

- Aprendizaje estructural de redes bayesianas mediante diferentes algoritmos de aprendizaje estructural.
- Aprendizaje paramétrico mediante la estimación por máxima verosimilitud.
- Calculo para métricas globales, y validación cruzada en diferentes variantes.
- Interfaz gráfica de usuario para inspeccionar las RB.
- Permite evaluar el rendimiento de los clasificadores bayesianos.

Sin embargo, dada su complejidad y su capacidad de ser genérico, es del criterio de ser una herramienta difícil de comprender y manejar, con usabilidad bastante pobre.

Sumado a lo anterior, las potencialidades de Weka se expresan en la presentación de un marco para la generación y comparación de estas técnicas de aprendizaje automatizado, utilizando métricas de evaluación predefinidas y no seleccionables de forma independiente, pero una vez generado un modelo, específicamente un clasificador bayesiano, no permite asignar una clasificación a un nuevo caso ni su permanencia posterior en el conjunto de estudio [32], limitando su uso solo a la generación y evaluación parcial de los clasificadores bayesianos.

4.7.1. API de WEKA

WEKA presenta una API [32] que le proporciona a los programadores un alto nivel de interoperabilidad, integración y reusabilidad de las funcionalidades de la plataforma, tanto las disponibles en los distintos interfaces gráficos o a través de la consola, con el objetivo de extenderla, o para integrar funcionalidades ya implementadas y probadas en proyectos independientes, a través de código Java. Esta se encuentra documentada con una completa especificación del código java en línea, y una amplia colección de ejemplos.

La misma está constituida por 10 paquetes generales que a su vez contienen subpaquetes. A continuación se relacionan alguno de ellos:

- **`weka.core:`** Contiene las clases e interfaces que conforman el núcleo de Weka, por lo que su utilización es común en los distintos algoritmos implementados en WEKA.
- **`weka.classifiers:`** Paquete con las implementaciones de algoritmos de clasificación (tanto a métodos de clasificación discreta como de predicción numérica).
- **`weka.gui`**: Paquete con la implementación de los interfaces gráficos de WEKA.

Dentro de las funciones disponibles en la API de Weka se encuentran:

- Trabajo con estructuras de datos utilizando las clases básicas: **`Instance,`**

Attribute e Instante.

La clase **instance** define una estructura de datos basada en filas y columnas, los atributos (**weka.core.Attribute**) se corresponden a las columnas y las filas contienen un conjunto de instancias o ejemplos (**weka.core.Instance**) que se almacenan conteniendo los valores de sus atributos.

- Operaciones de escritura y lectura de datos.

Permite realizar funciones de entrada/salida con ficheros de datos en formato ARFF [32] disponibles en **weka.core.converters.ConverterUtils.DataSource.**

- Opciones para la generación y evaluación de un clasificador.

Todos los algoritmos de clasificación en Weka son ubicados en el paquete **weka.classifiers** y heredan de la clase **Classifier.** Para entrenar el clasificador con un juego de datos brinda la opción **buildClassifier(Instances data).**

El paso de la evaluación, incluye una la colección de parámetros estadísticos que se pueden configurar en la clase **weka.classifiers.Evaluación**, proporcionando el método **evaluateModel(Classifier c, Instances data)** para la evaluación con un juego de datos dedicado y el método **crossValidateModel(Classifier c,Instance data, Integer numFolds, new Random(Interger seed))** para realizar la validación cruzada sin distribuir. Para proporcionar los resultados de la evaluación, existen implementados varios métodos como **toSummaryString()** y **toClassDetailsString().**

- Permite incluir un nuevo algoritmo de aprendizaje estructura para un clasificador bayesiano mediante un proceso claramente especificado, definido por los siguientes pasos:

1. Selección de la clase de base (**SearchAlgorithm**).
2. Selección del paquete donde va a guardarse la implementación.
3. Creación de la clase e implementación (definir los parámetros del algoritmo como atributos de la clase e implementación del método: **buildStructure(BayesNetbayes Net, Instances instances)**).

4. Implementación y configuración de los métodos para la visualización y utilización del nuevo algoritmo desde la interfaz gráfica de Weka.

- Especificación de las diferentes opciones.

Para definir las opciones de cualquiera de los algoritmos se provee la clase **`weka.core.Option`**, permitiendo construir cualquier opción dinámicamente, estableciendo para ello un nombre, una descripción y la cantidad de argumentos.

- Visualización de la red

El gráfico generado a partir del aprendizaje de un clasificador bayesiano puede ser visualizado utilizando la clase **`weka.gui.graphvisualizer.GraphVisualizer`**. Esta clase puede visualizar la estructura de la red en formato XML BIF.

Además Weka cuenta con el Editor de Redes Bayesianas, que constituye una aplicación independiente. A pesar de presentar opciones más completas con respecto a la edición de la red, la manipulación y organización de los nodos, y la presentación de la TPC, la misma no está integrada directamente a la salida de un clasificador para visualizar la RB después de su generación. Este componente se encuentra implementado en la clase **`weka.classifiers.bayes.net.GUI`**.

En el contexto de la investigación realizada por los investigadores de la UCLV y las ventajas de utilizar Weka se decidió para probar los nuevos algoritmos que se diseñaron, realizarle extensiones a esta plataforma.

5. Técnicas de integración de software

El desarrollo e integración de software es la adaptación y mejora de lo que otros han realizado, pero buscando nuevos enfoques que cubran los retos de los usuarios. También está enfocado a como dos o más software pueden compartir procesos, información e interfaces de usuario, a partir de estándares y mecanismos bien definidos para la comunicación.

En este sentido, la temática está cobrando gran auge en el contexto de la integración de aplicaciones empresariales la cual se caracteriza por una amplia gama de sistemas de información heterogéneos (esto es, aquellos fundamentados en equipos y

tecnologías de software heterogéneas) que es necesario integrar, para lo cual se han definido varios enfoques como EAI (Enterprise Application Integration) [33], B2BI (Business to Business Integration),EII (Enterprise Information Integration) y ETL (Extract, Transform and Load) [34], entre otras.

Paralelamente al desarrollo e investigación de los mecanismos de integración que satisfagan el ámbito empresarial, se han definido, para este y otros contextos, diferentes modelos de integración guiados por la organización estructural en capas que presenta un software. De esta forma, la comunicación con una aplicación, en el mundo de la integración, se hace por medio de una o más de sus capas, por ejemplo, la base de datos, interfaz gráfica de usuario, lógica de negocio, etc., lo cual no entra en contradicción con la integración definida por niveles expuesta anteriormente.

Un modelo de Integración [34] [35] no es más que una definición de cómo las aplicaciones serán integradas, mediante la especificación de la naturaleza y los mecanismos empleados en el proceso. A continuación se exponen los tres modelos de integración principales:

- *Modelo de Integración de Presentación.*

Se basa en el acceso a las aplicaciones a través de su lógica de presentación (ver Fig.4), desarrollando de una nueva interfaz de usuario donde cada interacción en la nueva interfaz debe mapearse en la interfaz original.

Desde el contexto de la EIA, este tipo de integración se presenta como el nivel de integración usando interfaces de usuario gráficas (GUI) y aborda el problema de obtener la información necesaria para un operador humano independientemente del sistema que la mantenga. Para dos aplicaciones o sistemas heterogéneos, S1 y S2, es posible lograr su integración a un nivel simplemente visual a través del uso de interfaces gráficas.

Al definir una reutilización total de los sistemas a integrarse a través de interfaces de usuario, este modelo permite que sea empleado un menor esfuerzo en el desarrollo respecto a la implementación de un nuevo sistema desde cero. Debe usarse cuando se desea una nueva interfaz gráfica para aplicaciones existentes o una interfaz que

unifique las funcionalidades de varias aplicaciones. Tiene como ventajas una integración rápida y sencilla, aunque está más limitado pues las aplicaciones integradas restringen el acceso a los datos.

- *Modelo de Integración de Datos.*

Se basa en el acceso directo a los datos empleados por las aplicaciones (ver Fig.4), realizando una derivación de las capas de presentación y lógica de negocio existentes. Al proponer una integración de las herramientas con un modelo de integración de datos, las aplicaciones existentes anteriormente se mantendrían actualizadas, al contrario de lo que pasaría si se desarrollara un nuevo sistema desde cero, con el cual se trabajaría y se actualizarían solo el nuevo diseño de datos, dejando sin actualización a las aplicaciones anteriores.

En este tipo de integración se suelen utilizar herramientas y/o middleware de acceso a bases de datos:

- ODBC/JDBC: estándar que permite el acceso a cualquier base de datos que lo soporte (RF)
- Middleware de acceso a BD: está enfocado hacia el acceso a bases de datos
- Servicios de transformación de datos (ETL): permiten ofrecer una visión uniforme de la información.

Este modelo debe usarse cuando se desea combinar datos de varias fuentes, se provee acceso a múltiples aplicaciones que desean acceder a una misma fuente o se extraen datos de una fuente para su inserción en otra, con un formato diferente. Como ventaja presenta que permite reescribir de lógica de negocio, es fácil de implementar con herramientas y middleware, posibilitando el acceso a todos los datos, aunque resulta dependiente del modelo de datos.

- *Modelo de Integración Funcional.*

La integración funcional (ver Fig.4) consiste en la integración a nivel de lógica de negocio, siendo esta la parte más importante en cualquier sistema. En relación a este aspecto, los modelos de integración explicados anteriormente no son suficientemente flexibles porque no se puede acceder a la lógica de negocio y es necesario reconstruirla.

En la EIA este modelo se enfoca como el nivel de integración de procesos, donde dos o más aplicaciones pueden compartir su lógica de negocios (procesos) y sus datos. Esta manera de integrar aplicaciones se da directamente a través de sus procesos, o sea, el punto de integración está en el código de la aplicación original, las cuales interoperan mediante interfaces de programación (APIs) o mediante el pase de mensajes remotos entre objetos, posibilitando que los nuevos desarrollos se realicen teniendo como basamento el código original. La integración mediante interfaces de programación requiere que cada aplicación disponga de un conjunto de APIs, a través de los cuales otras aplicaciones pueden acceder a los procesos y datos que ella está dispuesta a compartir o hacer visible. La integración mediante pases de mensajes remotos emplea varios mecanismos tales como: Remote Procedure Call (RPC), objetos distribuidos, Middleware Orientado a Mensajes (MOM),Servicios Web, Monitores de procesamiento de transaccionales (TPMs) y Servidores de Aplicaciones basados en Componentes (SAC) [35] [36] [37].

La integración funcional debe usarse cuando se desea alguna de las opciones siguientes:

- Un nuevo interfaz de usuario (ahora se accede al código fuente).
- Acceso a datos a través de funcionalidad existente.
- Una acción se lleva a cabo dependiendo de lo que pase en otras aplicaciones.
- Se desea implementar un proceso de negocio que se utiliza desde múltiples aplicaciones.

Este tipo de integración presenta ventajas respecto a que es más robusta y flexible que las anteriores, lo cual implica una alta complejidad ya que integra a nivel de lógica de negocio y resulta difícil de implementar si no se dispone de código fuente, APIs o interfaces de comunicación.

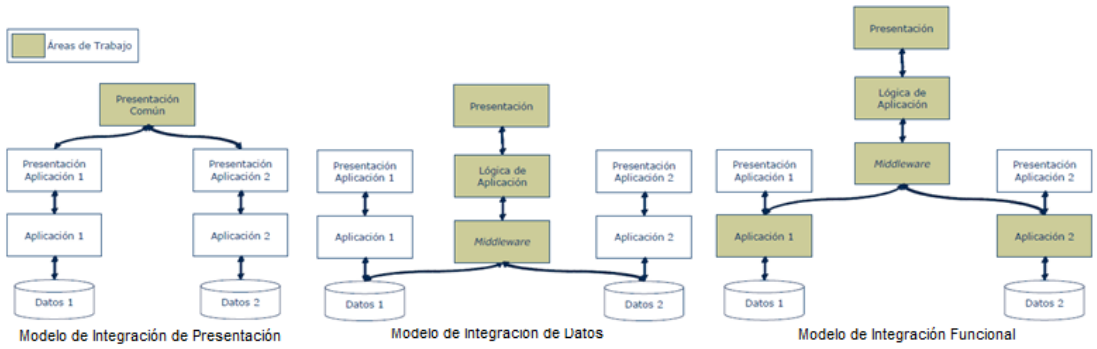


Fig.4 Modelos de integración.

De forma general, al hablar de integración, se pueden que tener en cuenta algunas restricciones (definidas para las empresas pero que pueden ajustarse a otros dominios de integración), para que una solución de integración sea viable:

- Después de realizar la integración, las aplicaciones involucradas existentes no deben cambiar. Un cambio en una de estas aplicaciones podrá afectar profundamente o hasta invalidar totalmente otras soluciones de integración, o incluso, los procesos de negocio que soportan esas aplicaciones.

- Después de integradas, las aplicaciones existentes antes de la integración deben mantenerse desacopladas las una de las otras como antes de la integración. La solución de integración no debe cambiar las aplicaciones involucradas generando dependencias en ellas que antes no existían.

CONCLUSIONES

En el presente capítulo se expusieron los principales conceptos desde el punto de vista matemático que permiten definir los procesos de generación, evaluación y uso de los clasificadores bayesianos, lo que permitió obtener un entendimiento preciso de los mismos para su integración. Además se realizó un estudio de las principales aplicaciones informáticas que dan soporte a estas operaciones de forma independiente. Partiendo del análisis anterior, y las bondades que ostenta la reutilización del código de la plataforma Weka, es posible orientar el desarrollo de la nueva aplicación hacia la extensión e integración de las funcionalidades que presenta Weka, lo cual debe garantizar una disminución sustancial del tiempo de desarrollo de un prototipo de un software a la medida.

Finalmente se expusieron los principales conceptos y paradigmas de la integración de software, lo cual permitió guiar el desarrollo de la investigación en función del Modelo de Integración Funcional.

CAPÍTULO 2. DESARROLLO DEL SISTEMA

Para la definición del nuevo sistema, partiendo de la extensión e integración de las funcionalidades de Weka, se listan los requisitos funcionales y se realiza una comparación entre estos y las funcionalidades expuestas por la biblioteca de clases a reutilizar. Además, en el presente capítulo, se especifica la organización estructural del sistema a través de la arquitectura del mismo y la presentación del diseño e implementación de componentes específicos.

1. Requisitos del sistema

1.1. Requisitos funcionales

RF 1. Realizar el aprendizaje estructural de clasificadores bayesianos.
RF 1.1 Realizar el aprendizaje estructural de clasificadores bayesianos incluyendo los algoritmos de aprendizaje estructural optimizados para ambientes bioinformáticos y biomédicos ByNet, BayesChaid y BayesPSO.
RF 1.2 Permitir seleccionar los parámetros de entrada de los algoritmos incluyendo el estadígrafo Chi cuadrado a utilizar (Pearson, Corregido de Yates, y Mantel y Haenszel) para los algoritmos ByNet y BayesChaid.

RF 2. Realizar el aprendizaje paramétrico partiendo del aprendizaje estructural de los clasificadores bayesianos generados.

RF 3 Mostrar gráficamente la red bayesiana generada.
RF 3.1 Permitir mover los nodos de la red bayesiana generada.
RF 3.2 Permitir organizar los nodos de la red bayesiana generada aplicando varias estrategias.
RF 3.3 Mostrar la tabla de probabilidad condicional de un nodo de la red bayesiana.
RF 3.4 Mostrar gráficamente los conglomerados de la red para la aplicación del algoritmo de inferencia de árboles de unión.

RF 4. Evaluar los clasificadores generados.

RF 4.1 Permitir registrar los parámetros de evaluación para su posterior uso en la comparación y selección del clasificador que más se ajuste al dominio de estudio.
RF 4.2 Seleccionar los parámetros de evaluación a aplicar. Entre los mismos se debe encontrar: rFN, rFT, rTP, rTN, ROC, Matthews CC, Exactitud, Precisión.
RF 4.3 Evaluar el clasificador utilizando el mismo fichero de entrenamiento como de prueba.
RF 4.4 Evaluar el clasificador utilizando un fichero de entrenamiento y uno de prueba.
RF 4.5 Evaluar el clasificador utilizando un porciento del fichero de entrenamiento como datos de prueba.
RF 4.6 Evaluar el clasificador a través de la operación de validación cruzada tanto en un entorno centrado como en un ambiente distribuido (T-Arenal).

RF 5 Comparar varios clasificadores.
RF 5.1 Mostrar los resultados de la evaluación de varios clasificadores en forma de tabla que permita resumir los parámetros y ordenarlos de acuerdo a la selección del usuario.
RF 5.2 Mostrar los resultados de la evaluación de varios clasificadores a través de gráficos.
RF 5.3 Aplicar la prueba de Friedman a varios experimentos.
RF 5.3.1 Seleccionar los experimentos a aplicar la prueba de Friedman.
RF 5.3.2 Seleccionar el parámetro de evaluación que se quiere comparar en la prueba de Friedman.

RF 6 Realizar inferencias con un clasificador bayesiano generado.
RF 6.1 Seleccionar un clasificador bayesiano para realizar inferencias a partir de los generados y disponibles por la aplicación.
RF 6.2 Cargar un clasificador bayesiano desde un fichero para utilizarlo en la clasificación de nuevos datos.
RF 6.3 Cargar una estructura de datos vacía relacionada con el clasificador bayesiano seleccionado para realizar inferencias.

RF 6.4 Cargar un fichero con datos como estructura de datos relacionada con el clasificador bayesiano seleccionado para realizar inferencias.
RF 6.5 Editar la estructura de datos cargada permitiendo insertar nuevos valores para las entidades a clasificar.
RF 6.6 Seleccionar la variable a clasificar.
RF 6.7 Clasificar las nuevas instancias a partir de los valores entrados utilizando el algoritmo de árboles de unión.
RF 6.8 Salvar la estructura de datos con los valores de las nuevas instancias clasificadas.

RF 7 Salvar en un fichero el clasificador bayesiano generado.
RF 8 Salvar el experimento realizado.
RF 9 Cargar un experimento desde un fichero.

RF 10 Visualizar y editar un conjunto de datos.
RF 10.1 Abrir fichero en formato ARFF.
RF 10.2 Insertar nuevas instancias y sus correspondientes valores.
RF 10.3 Adicionar nuevas instancias y sus correspondientes valores.
RF 10.4 Eliminar instancias y sus correspondientes valores.
RF 10.5 Eliminar atributos y sus correspondientes valores.

1.2. Requisitos no funcionales

RNF 1. Usabilidad.
La aplicación debe permitir un fácil y lógico acceso a todas las funcionalidades:

- Permitir visualizar y tener rápido acceso a los datos que representan el conjunto de estudio.

- Tener los menús de configuración y botones de ejecución de las operaciones en paneles que presenten la misma posición relativa a la ventana (extremo derecho de la ventana) independientemente de la funcionalidad que se esté ejecutando.

- Mostrar la salida de las operaciones en paneles que presenten la misma posición relativa a la ventana independientemente de la funcionalidad que se esté ejecutando.

- Los paneles anteriores deben poderse ocultar y mostrar de acuerdo a las necesidades del cliente.

- En la pantalla de clasificación de nuevas entidades se debe tener acceso tanto a los clasificadores generados anteriormente por la herramienta (presentes en la memoria) como a clasificadores guardados en ficheros.

RNF 2. Formato.
La aplicación debe ser capaz de cargar y trabajar con ficheros en los siguientes formatos:

- XML BIF (Interchange Format for Bayesian Networks) [32] para la estructura de la red bayesiana.
- ARFF (Attribute-Relation File Format) [32] para la estructura de datos.
- *.model* para los ficheros que contienen clasificadores generados.

RNF 3. Portabilidad
La aplicación debe ser capaz de ejecutarse tanto sobre plataformas libres GNU/Linux, como en plataforma privativa Windows XP/Vista/7.

RNF 4. Software.
Se requiere para el funcionamiento del sistema disponer de Java VM 1.4 o posterior.

FRN 5. Presentación del idioma.
El software debe implementarse de forma que permita la internacionalización.

2. Funcionalidades de Weka a reutilizar

Después del análisis de las funcionalidades aportadas por la API de Weka en el Capítulo 1, se puede realizar una comparación entre las mismas y los requisitos

funcionales establecidos (Tabla 1), obteniendo conclusiones respecto a su reutilización o extensión.

Funcionalidad	*Implementada en la API de Weka*	*Parcial (P) o totalmente (T)*	*Observaciones*
RF 1	Si	P	
RF 1.1	Si	P	Falta incluir los nuevos algoritmos.
RF 1.2	Si	P	Tiene los mecanismos para incluir parámetros, pero no está implementado el correspondiente a Chi-cuadrado.
RF 2	Si	T	
RF 3	Si	T	A pesar de que no está implementado en la visualización del gráfico cuando se generan y evalúan los clasificadores bayesianos, estas funcionalidades se encuentran implementadas en el Editor de Red Bayesiana.
RF 3.1 al 3.4	Si	T	
RF 4	Si	P	
RF 4.1	Si	T	
RF 4.2	Si	P	No permite seleccionar independientemente las métricas a utilizar. No tiene implementado las métricas Exactitud y Matthews CC, ni la prueba no paramétrica de Friedman.
RF 4.3 al 4.5	Si	T	
RF 4.6	No		No permite distribuir la operación de validación cruzada en un ambiente de controlado de cálculo distribuido (T-Arenal).
RF 5	No		
RF 5.1	No		Muestra los resultados de la evaluación como texto plano.
RF 5.2	No		
RF 5.3	No		
RF 6	No		Tiene implementado el algoritmo para el proceso de la evaluación de los clasificadores desde la interfaz visual, pero no permite insertar nuevos casos a un conjunto de datos y clasificarlos.
RF 6.1 al 6.6	No		

RF 6.7	Si	P	Tiene implementado el algoritmo de árboles de unión.
RF 7	Si	T	
RF 8 **RF 9**	No		El experimento se define como una entidad propia del negocio de la nueva aplicación.
RF 10	Si	P	A pesar de que no está implementado conjuntamente con la interfaz de generación y evaluación de RB, Weka tiene un componente independiente que permite visualizar y editar parcialmente datos en formato ARFF.
RF 10.1	Si	T	
RF 10.2 y 10.3	No		
RF 10.4 y 10.5	Si	T	

Tabla1. Comparación de Requisitos funcionales con respecto a funcionalidades propiciadas por Weka.

A partir del análisis de la tabla anterior se observa que existen:

- 8 funcionalidades presentes de forma total que se pueden reutilizar.
- 7 funcionalidades presentes de forma parcial que se pueden extender y reutilizar.
- 9 funcionalidades no presentes.

Esta conclusión influyó decisivamente en las determinaciones llevadas a cabo desde el punto de vista arquitectónico y de diseño para la implementación del sistema.

3. Arquitectura del sistema

La Arquitectura del Software representa la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán. La misma se ilustra a través de la vista lógica del sistema propuesta en la Fig.5, donde se muestran las clases representativas y su organización en paquetes y subsistemas.

La arquitectura general del sistema se definió a partir del patrón arquitectónico en capas, lo cual permite segmentar la lógica de la solución en tres elementos:

- Capa de presentación: Constituye la interfaz del usuario con el sistema, y se comunica únicamente con la capa de negocio.

- Capa de negocio: Es donde se ejecutan las funcionalidades y reglas que deben cumplirse desde el punto de vista del negocio. Se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados de las peticiones del usuario, y con la capa de datos, para consultar la información o guardar la misma.
- Capa de datos: Permite obtener la información persistente del sistema.

En la vista lógica del sistema (Fig.5) se presentan las clases de diseño más significativas identificadas para implementar las funcionalidades del sistema, las cuales han sido agrupadas en tres paquetes según su funcionalidad, en representación de las capas presentes en patrón arquitectónico expuesto anteriormente. En la misma, además se muestran las dependencias de uso de los diferentes subsistemas de diseño, señalándose los elementos de integración a partir del uso de la API de Weka.

Dado el diagrama mencionado anterior (Fig. 5) se explicarán en las siguientes secciones los elementos más representativos del diseño e implementación del sistema.

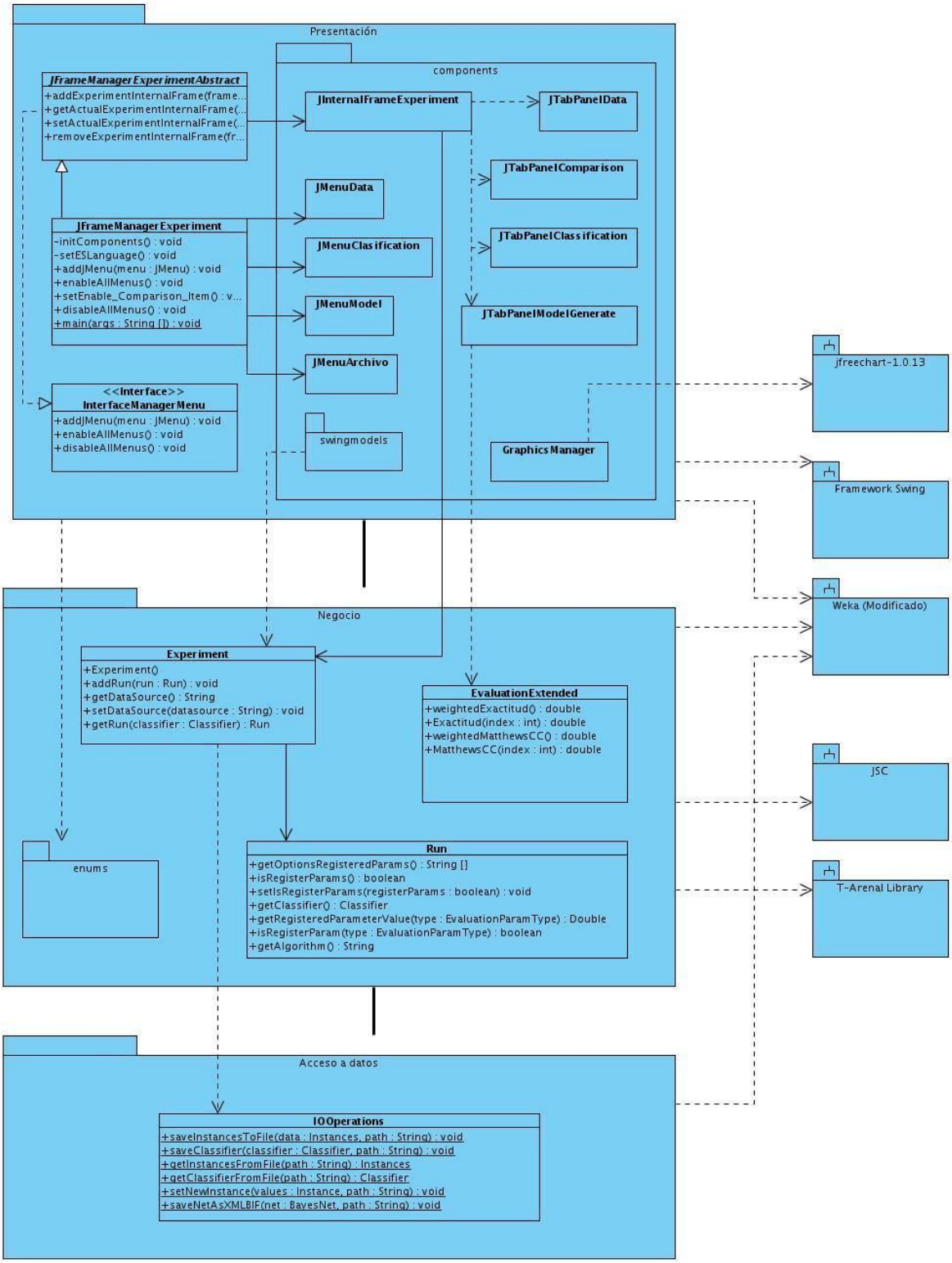


Fig.5 Vista lógica del sistema.

4. Elementos del diseño e implementación del negocio del sistema

4.1. Adición de los algoritmos ByNet, BayesChaid y BayesPSO a la plataforma Weka V 3.6.3

Para la adición de los nuevos algoritmos se realizaron los siguientes pasos:

a. Descompilación (obtener las clases *.java* a partir de los ficheros *.class*) de estos algoritmos presentes en la biblioteca de clases base WekaParallelWeka V 3.5.6.
b. Revisión y corrección de los ficheros *.java* resultantes del proceso de descompilación.
c. Adición e implementación del cálculo para el parámetro Chi-cuadrado a los algoritmos ByNet y BayesChaid.
d. Integrar las clases y los paquetes correspondientes a Weka v3.6.3.
e. Configurar la búsqueda automática de clases en Weka para la visualización de los nuevos algoritmos en la GUI, a través de la configuración del recurso **`weka.gui.GenericObjectEditor.props.`**
f. Compilación y compresión de la nueva biblioteca de clases *.jar* para su utilización en la implementación del sistema.

4.2. Extensión de los algoritmos ByNet y BayesChaid con la implementación del parámetro Chi-cuadrado

Los algoritmos ByNet y BayesChaid utilizan la prueba Chi-cuadrado para buscar las variables más significativamente relacionadas con la variable dependiente [24]. Estos fueron diseñados con la implementación de Chi-cuadrado de Pearson, sin embargo, esta prueba presenta otras variantes de cálculo, entre las que se encuentran Chi-cuadrado Mantel y Haenszel y Chi-cuadrado Corregido de Yates. Para ello se llevaron a cabo las siguientes tareas:

a. Definición de un enumerador para los tipos de Chi-cuadrado establecidos en **`enum EnumEstadigrafoType.`**
b. Inserción de un nuevo atributo en la clase ByNet y BayesChaid que recogiera el valor en cuestión, y la implementación de los métodos asociados: **`enumEstadigrafoType m_nEstadigrafo`**

```
selectedTagget getEstadigrafo(){...}
string estadigrafoTipText(){...}
void setEstadigrafo(SelectedTagnew Estadigrafo) {...}.
```

c. Definición de una nueva opción, '-e', para entrar este parámetro, y se redefinición de los métodos correspondientes:

```
public void setOptions(String options[])
public String[] getOptions()
public Enumeration listOptions().
```

d. Implementación del cálculo correspondiente redefiniendo el método `private double chaidWeight(...).`

4.3. Integración de la evaluación del clasificador de la API de Weka y su extensión dentro del sistema

Para implementar las nuevas métricas y poder tener acceso a las definidas en Weka se decide heredar de la clase **`weka.classifiers.Evaluation,`** y extender la misma desde el sistema a través de la herencia. En esta nueva clase se añadieron los métodos que implementen los cálculos correspondientes a los parámetros Exactitud y Matthews CC, manteniendo el estándar de nombres definido para los métodos de este tipo por la clase padre (Fig.6).

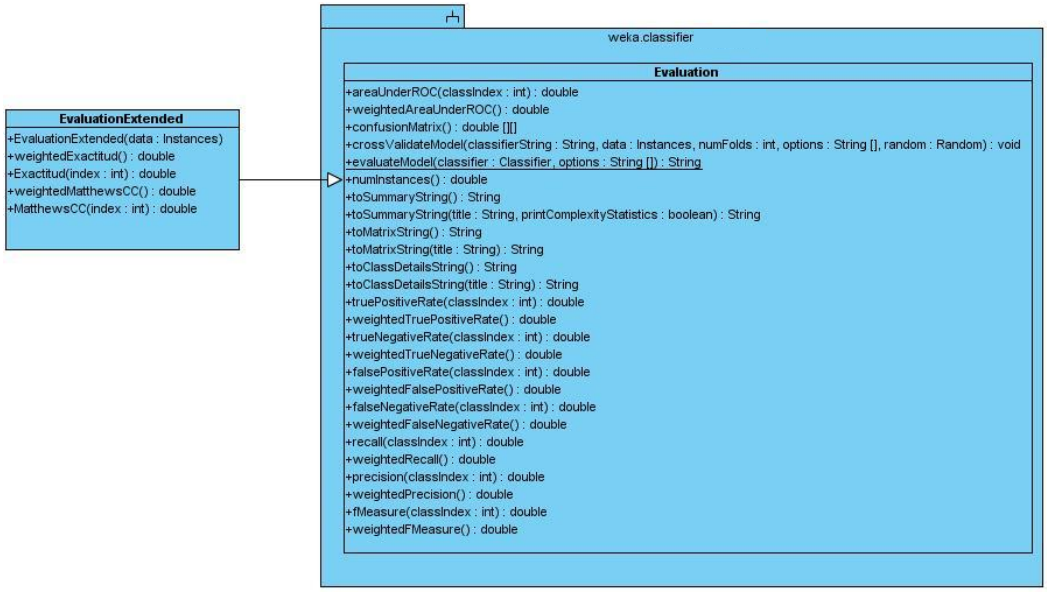


Fig.6 Diseño de clases relacionados con la evaluación de un clasificador bayesiano.

4.4. Implementación de la prueba no paramétrica de Friedman.

Para la implementación de las prueba no paramétrica de Friedman se utilizó la biblioteca de clases Java Statistical Classes [38] que presenta implementada la funcionalidad mediante el método

`FriedmanTest(double[][] values, String[] blockLabels, String[] treatmentLabels).`

El parámetro *blockLabels* se corresponde con el número de filas de la tabla de comparación, en este caso serán los experimentos realizados definidos en correspondencia con un conjunto de datos diferente. El parámetro *treatmentLabels* se corresponde con la identificación de los distintos algoritmos de aprendizaje estructural con los cuales se generó un clasificador bayesiano en cada experimento, y finalmente el parámetro *values,* contiene los valores de determinada métrica de evaluación objetivo a comparar mediante la prueba de Friedman.

Como antesala a la llamada de este método se implementó un algoritmo que dado los experimentos a comparar y la métrica por el usuario, selecciona la unión de todos los posibles algoritmos de aprendizaje estructural que registraron esta métrica y se corrieron en los experimentos seleccionados, logrando así una prueba más integral, con mayor cantidad de datos. En este sentido es válida la implementación futura de un algoritmo que permita por parte de los especialistas seleccionar también los posibles algoritmos a combinar en la prueba.

5. Elementos del diseño e implementación de la interfaz visual del sistema

5.1. Extensión de las funcionalidades del Editor de Redes Bayesianas de Weka

Se redefinió el componente **`weka.classifiers.bayes.net.GUI`** añadiéndole métodos y la lógica pertinente para su adecuación a los RF definidos para su integración con el sistema, con el objetivo de mostrar la RB obtenida del aprendizaje estructural. Las acciones efectuadas se especifican a continuación (ver Fig.7):

a. Redefinición de la clase base **`MyAction`** para permitir la internacionalización

b. Inserción de cuatro botones al panel: *Imprimir, Exportar, VisualizarConglomerados, VisualizarProbabilidadesMarginales.*

c. Adición de dos clases de tipo **`javax.swing.AbstractAction`** para definir la acción de los botones insertados *VisualizarConglomerados y VisualizarProbabilidadesMarginales*:

```
classActionShowMargins extends MyAction{…}
classActionShowCliques extends MyAction{…}.
```

d. Implementación de un método que permita al panel representar la red visual a partir de la descripción de la red en formato BIFXML

```
public void readBIFFromString(String BIFString).
```

e. Redefinición del menú desplegable a partir de la selección de un nodo, restringiendo algunas funcionalidades, y su internacionalización.

f. Exposición pública de la llamada al método layoutGraph() para posibilitar su acceso desde otras clases.

Estos cambios permitieron adicionar nuevas funcionalidades opcionales a Weka, sin que perdiera su definición e independencia original.

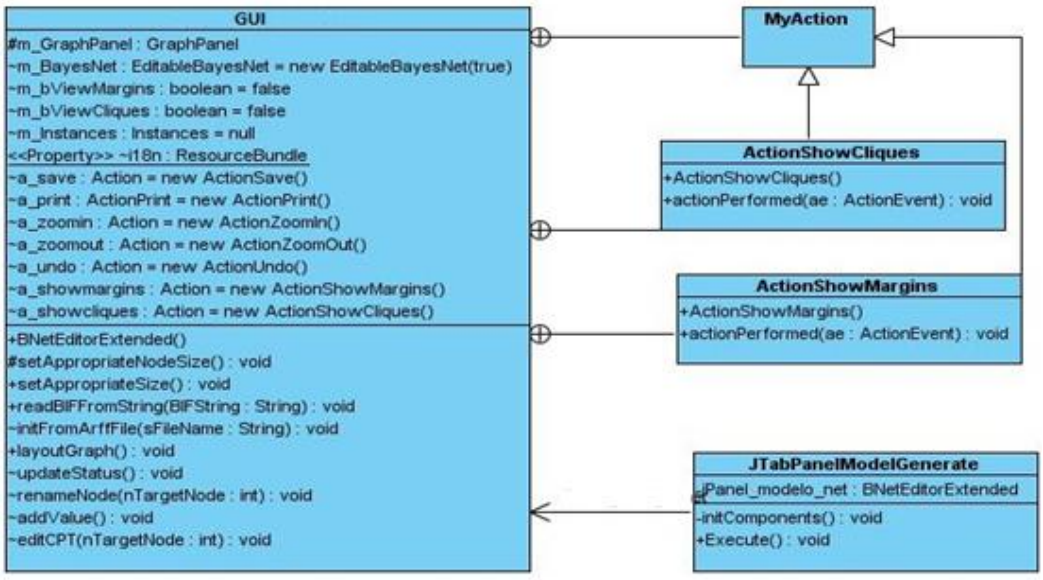


Fig.7 Diagrama de clases de diseño para la redefinición del Editor de Redes Bayesianas de Weka.

5.2. Extensión de las funcionalidades del panel de selección de los algoritmos de aprendizaje estructural en Weka

Se redefinió la clase **`weka.gui.PropertySheetPanel`** insertando la implementación de dos nuevos métodos (**`setBorderName(String name)`** y **`setHelpButText(String text)`**) para el establecimiento del encabezado y el nombre del botón de información del algoritmo (predefinidos en inglés) de forma opcional, tal que se puedan redefinir de acuerdo al lenguaje que tenga la aplicación que esté utilizando este panel de Weka. En el caso específico del sistema desarrollado en la investigación, la internacionalización es un RNF.

La clase anterior es utilizada por **`weka.gui.GenericObjectEditor`**, encargada de mostrar el panel de selección de los algoritmos para el aprendizaje estructural de la RB y visualizar dinámicamente los parámetros del mismo. Para su reutilización, fue necesario insertar la implementación de dos nuevos métodos **`setButtonChooseName(String name)`** para establecer de forma opcional el texto del botón de selección, y **`setVisibleButtons(boolean visible)`** para establecer si se quiere mostrar o no el panel de botones inferior de este componente.

Estos cambios permitieron que Weka fuera modificada de acuerdo a las necesidades del sistema, aportando nuevas funcionalidades opcionales, sin perder su definición y autonomía anterior.

5.3. Definición de las clases visuales del sistema

Las clases de la interfaz visual se definieron utilizando el framework Swing de Java. La pantalla principal (JFrameManagerExperiment) funciona como un gestor de marcos o ventanas internas (JInternalFrameExperiment) que permite adicionar, mover y cerrar estas. Cada ventana interna se corresponde desde el punto de vista del negocio a un *experimento*. Un experimento puede utilizar solo un conjunto de datos (JTabPanelData) y tantos paneles de generación y evaluación de redes bayesianas (*escenarios del experimento*) (JTabPanelModelGenerate) como entienda conveniente el especialista, así como pestañas de clasificación (JTabPanelClassification). Las visualizaciones

respecto a la comparación (JTabPanelComparison) de la evaluación de varios clasificadores son únicas para cada experimento. (Ver Fig.5 *Paquete Presentación*)

Para el diseño basado en la reutilización de componentes, contribuyendo con el bajo acoplamiento y alta cohesión, los componentes visuales del sistema (ventanas, pestañas, marcos, menús, paneles, y modelos de componentes como tablas y listas), se implementaron como clases independientes definiendo para la mejor organización y entendimiento del código una plantilla de clase. La misma define el método **`initComponents()`** para instanciar y configurar los elementos visuales del componente y el método **`setActionsListeners()`** para la asignación e implementación de los métodos de escucha de eventos para todos los elementos del componente definidos en la clase.

CONCLUSIONES

Partiendo de la lista de los requisitos funcionales del sistema y su comparación con las funcionalidades expuestas por la biblioteca de clases Weka, se determinó que ocho funcionalidades estaban presentes de forma completa y siete en forma parcial, de un total de 24 RF. Lo anterior confirmó la necesidad de realizar un diseño e implementación orientados hacia la extensión e integración de las funcionalidades de Weka. Además en el capítulo se mostró la vista lógica del sistema donde se aprecia la aplicación del patrón arquitectónico en capas, de forma que el sistema pueda aprovechar las ventajas del mismo.

En el capítulo se presentan las descripciones de las modificaciones realizadas a Weka a través de diagramas de clase de diseño y fragmentos de código, mostrando desde el punto de vista de diseño como aprovechar al máximo sus funcionalidades, y proveer otros métodos necesarios para la integración con el sistema sin afectar su funcionamiento básico. Lo anteriormente expresado permitió la integración de los procesos de generación, validación y uso de clasificadores bayesianos, y constituye una guía para otros proyectos independientes que necesiten reutilizar funcionalidades ya implementadas en otras aplicaciones, específicamente en Weka.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN

Las pruebas de software, son los procesos que permiten validar y revelar la calidad de un producto software. Para desarrollar las mismas se han definido diferentes niveles de prueba, técnicas y métodos. En el presente capítulo se realiza la validación del sistema mediante pruebas unitarias, pruebas de integración y pruebas de usabilidad. Además se expone un breve manual de usuario del sistema.

1. Pruebas unitarias del sistema

Las pruebas unitarias de software constituyen un nivel de prueba enfocado a comprobar el correcto funcionamiento de un módulo (unidad) del sistema. Su objetivo es aislar cada parte del programa y mostrar que las partes individuales son correctas. Las mismas deben cumplir ciertos requisitos:

- Automatizable: el uso de sistemas o elementos de software para controlar las pruebas permite una mayor gestión de las mismas y su reutilización.
- Repetibles o reutilizables: deben crearse pruebas que puedan ser fácilmente ejecutadas en más de una ocasión.
- Independientes: la ejecución de una prueba no debe afectar a la ejecución de otra.

Desde el punto de vista del paradigma de la programación Orientado a Objetos, el concepto de pruebas unitarias es más versátil. En este contexto, una unidad puede ser desde un módulo del sistema, un submódulo, un componente, o una clase u objeto encapsulado, siendo estos últimos la menor unidad a probar.

Weka provee varios mecanismos para realizar pruebas unitarias a las nuevas clases que se incluyen en su API. Los mismos permiten hacer pruebas sobre elementos específicos de las clases como el establecimiento correcto de las opciones, o la visualización de parámetros desde la GUI de Weka.

Las pruebas unitarias del sistema se realizaron desde la línea de comandos, sobre la biblioteca de clases Weka modificada (*wekam.jar*), compilada con los algoritmos de aprendizaje estructural ByNet, BayesChaid y BayesPSO implementados.

1.1. Pruebas sobre el manejo de opciones

El manejo de opciones de los algoritmos de aprendizaje estructural implementados puede ser probado utilizando la clase **`weka.core.CheckOptionHandler`** a través de la línea de comandos:

`weka.core.CheckOptionHandler –W *clase_del_algoritmo* –D.`

Entre las opciones disponibles para este tipo de prueba se van a utilizar:

`-D Activar depuración de la salida`

`-W Activar la salida con el nombre completo de la clase analizada.`

La prueba realizada para los tres algoritmos, en todos los casos devolvió resultados positivos. A continuación se muestran (ver Fig.8) los resultados de la ejecución de la prueba para la clase que implementa el algoritmo ByNet.

```
ema@lp04-402-19:~$ java -cp wekam.jar weka.core.CheckOptionHandler -W weka.classifiers.bayes
.net.search.deterministic.ByNet -D
OptionHandler: weka.classifiers.bayes.net.search.deterministic.ByNet

--> Info
Default options: -A 0.05 -D 3 -T 10 -I 30 -e Pearson
User options:
--> Tests
ListOptions...
-A <Significance [0..1]>
        Max ChiSquare Significance used in search

-Z <nr of Trees>
        Maximum number of Trees

-D <nr of levels>
        Maximum Depth

-D <nr of Instances>
        Minimun instances to Split

-V
        Habilitar modo interactive

-e <ChiSquareStatistic>
        Estadígrafo

yes
SetOptions...yes
Default options...yes
Remaining options...
  remaining:
yes
Canonical user options...
  Getting canonical user options: -A 0.05 -D 3 -T 3 -I 30 -e Pearsor
  Setting canonical user options
  Checking canonical user options
yes
Resetting options...
  Setting user options
  Resetting to default options
  Checking default options match previous default
yes
```

Fig.8 Salida en consola de la prueba al algoritmo ByNet.

1.2. Pruebas sobre la disponibilidad de las opciones en la GUI

Para verificar si todas las opciones definidas para los algoritmos están disponibles desde la interfaz visual de usuario (GUI) de Weka se utiliza la clase **`weka.core.CheckGOE.`** Además, esta clase verifica que estén implementados el método **`globalInfo()`**, genérico para cada algoritmo, y los métodos de ayuda correspondientes para cada parámetro. A continuación se muestran los resultados de la prueba para los tres algoritmos (ver Fig.9).

```
ema@lp04-402-19:~$ java -cp wekam.jar weka.core.CheckGOE -W weka.classifiers.bayes.net.search.deterministic.ByNet
Object: weka.classifiers.bayes.net.search.deterministic.ByNet

--> Tests
Global info...yes
Tool tips...yes
ema@lp04-402-19:~$ java -cp wekam.jar weka.core.CheckGOE -W weka.classifiers.bayes.net.search.deterministic.BayesCHAID
Object: weka.classifiers.bayes.net.search.deterministic.BayesCHAID

--> Tests
Global info...yes
Tool tips...yes
ema@lp04-402-19:~$ java -cp wekam.jar weka.core.CheckGOE -W weka.classifiers.bayes.net.search.global.BayesPSO
Object: weka.classifiers.bayes.net.search.global.BayesPSO

--> Tests
Global info...yes
Tool tips...no (missing: [VMaxTipText])
```

Fig.9 Salida en consola de la prueba: Disponibilidad de opciones en la GUI.

A partir de esta prueba se pudo detectar y corregir la implementación del algoritmo BayesPSO.

1.3. Pruebas unitarias del objeto de tipo clasificador

Weka permite utilizar las ventajas del framework de pruebas unitarias de Java, JUnit v4.5 para validar un clasificador. Con el objetivo de comprobar los clasificadores generados por los nuevos algoritmos, se utilizó la clase **`BayesNetTest`**, que se redefinió con la instanciación del clasificador asignándole como algoritmo de búsqueda un objeto de tipo ByNet, BayesChaid, o BayesPSO.

El resultado de esta prueba verifica 23 aspectos diferentes, y se ejecutó satisfactoriamente para todos los algoritmos.

2. Pruebas de integración

Las pruebas de integración tienen como objetivo identificar errores introducidos por la combinación de programas o componentes probados unitariamente, permitiendo validar el flujo de control entre los módulos, y sobre los datos que son intercambiados entre ellos.

Para comprobar la integración de los componentes de Weka con el sistema y de los módulos o pantallas típicas intra-sistema, se definieron cuatro casos de prueba, uno por cada pantalla típica. Estos casos de pruebas se especifican de acuerdo a los diferentes escenarios o flujos alternos que presenta el sistema, y están enfocados a verificar la adecuada correspondencia entre el flujo de comunicación y el paso de atributos de uno a otro módulo del sistema siguiendo el mapa de integración representado en la Fig.10.

De la realización de este tipo de pruebas se derivaron cinco no conformidades que fueron solucionadas exitosamente.

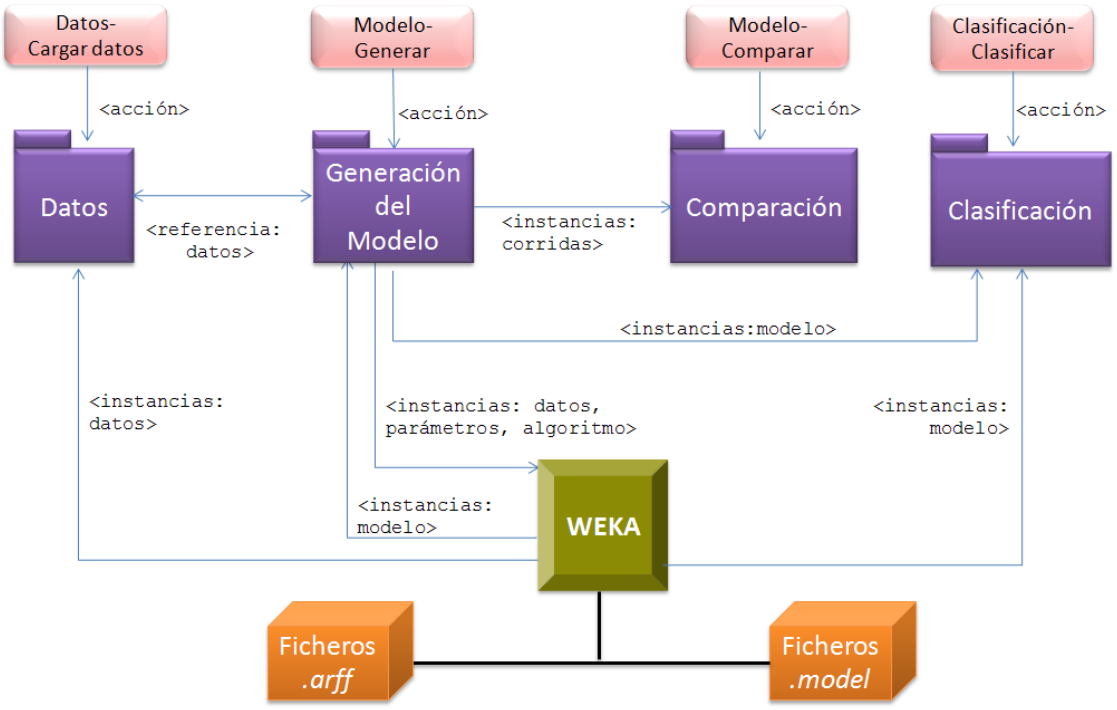


Fig.10 Mapa de integración del sistema.

3. Pruebas de usabilidad

La usabilidad, como concepto genérico en el mundo de desarrollo de software, se refiere a la claridad y elegancia con que se diseña la interacción del usuario con una aplicación informática. En las definiciones formales (ISO/IEC 9126 e ISO/IEC 9241), se destaca el papel del usuario y del contexto de uso como entes fundamentales en el concepto de usabilidad de un producto, ya que este ultimo sólo tendrá la capacidad de ser usado con efectividad en condiciones específicas y por usuarios particulares.

Para medir la usabilidad del sistema se aplicó el cuestionario de evaluación con 12 preguntas (ver Anexo 2) desarrollado para percibir el grado de uso y utilidad basado en el Modelo de Aceptación de Tecnología (TAM: Technology Acceptance Model) [39].
Además, para realizar las pruebas, los evaluadores desarrollaron 15 tareas, que cubren todas las funcionalidades que pueden ejecutarse con la herramienta, y se listan a continuación:

1. Seleccionar el juego de datos para generar el clasificador bayesiano.
2. Visualizar y editar un conjunto de datos para generar el clasificador bayesiano.
3. Establecer el tipo de prueba para la evaluación.
4. Registrar y seleccionar los parámetros de evaluación.
5. Seleccionar los parámetros de entrada de los algoritmos para la generación de la red bayesiana.
6. Generar y evaluar el clasificador bayesiano utilizando el algoritmo ByNet, BayesChaid o BayesPSO.
7. Visualizar gráficamente la red bayesiana generada.
8. Comparar varios clasificadores utilizando la tabla de comparación de métricas.
9. Comparar varios clasificadores seleccionando y utilizando los gráficos disponibles por la aplicación.
10. Comparar varios clasificadores aplicando la prueba de Friedman a varios experimentos.
11. Seleccionar un clasificador bayesiano para realizar inferencias a partir de los generados y disponibles por la aplicación, o desde un fichero.
12. Editar la estructura de datos relacionada con el clasificador seleccionado para realizar las inferencias, insertando nuevos valores para las entidades a clasificar.

13. Seleccionar el atributo a clasificar.
14. Clasificar las nuevas instancias a partir de los valores entrados.
15. Salvar la estructura de datos con los valores de las nuevas instancias clasificadas.

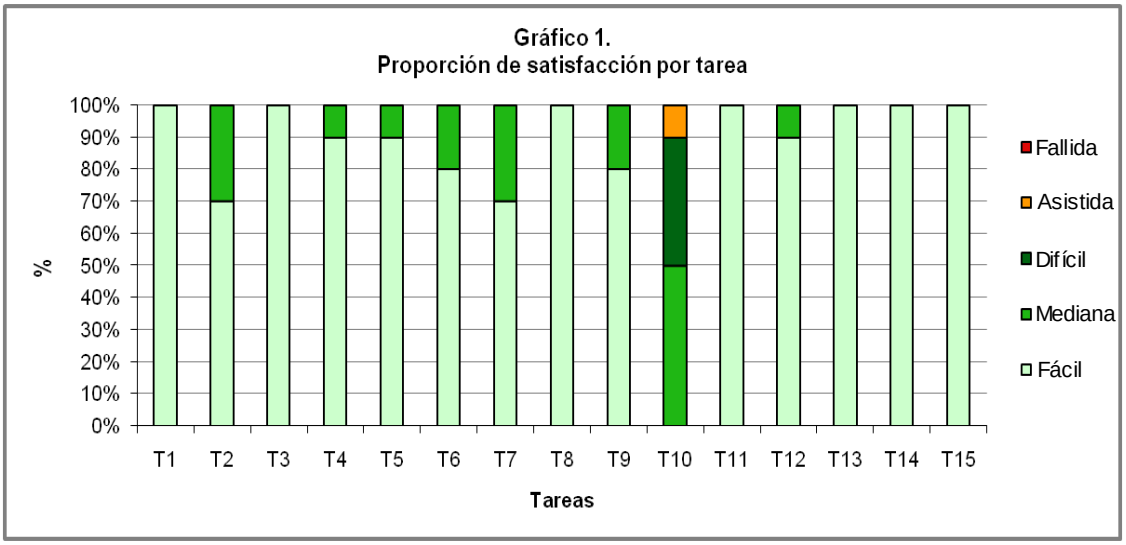
El objetivo de las pruebas se definió en base a las siguientes métricas:

- Satisfacción del usuario: Describe cómo valora el usuario su interacción con el sistema durante la ejecución de las tareas establecidas.
  Se define desde el punto de vista cualitativo mediante cinco criterios:
  Fácil: En la primera prueba no se presenta ningún problema.
  Mediana: De dos a tres pruebas realizadas se observan dificultades.
  Difícil: De tres a cuatro pruebas realizadas se mantienen las dificultades.
  Asistida: Se necesita asistencia técnica para realizar la prueba.
  Fallida: La prueba falla o el usuario desiste de llevar a cabo la ejecución de la misma.
  Desde el punto de vista cuantitativo se expresa a través de una escala de siete puntos [1,7], que representa el grado de confianza con el sistema que tiene el usuario al ejecutar cada tarea.
- Tareas ejecutadas satisfactoriamente por el usuario: Se define a partir de la cantidad de tareas ejecutadas satisfactoriamente por el usuario sin asesoramiento externo.
- Tiempo: Representa a el tiempo requerido para la ejecución de cada tarea.

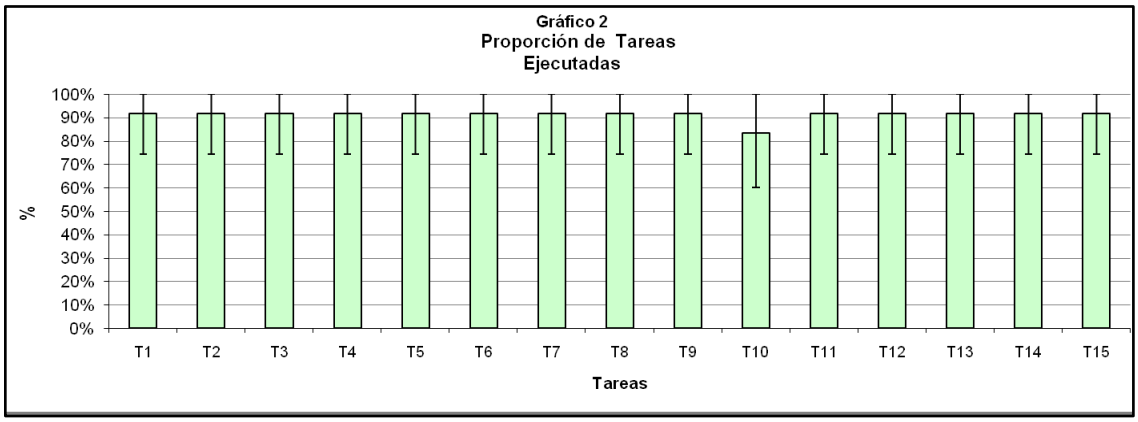
El grupo de evaluadores estuvo constituido por un total de diez personas, cinco de ellas especialistas en el uso de herramientas para la generación de redes bayesianas e investigadores y desarrolladores del tema, y el resto formado por especialistas, ingenieros e investigadores que llevan a cabo tareas generales de minería de datos. Los mismos expresaron su grado de acuerdo o desacuerdo con respecto a cada planteamiento del cuestionario después de la ejecución de las tareas en base a la escala de aceptación de siete puntos de Likert [40]. Las pruebas se automatizaron utilizando la herramienta UsabilityDataLogger v5.1.1.

Como resultado de las pruebas podemos apreciar en el Gráfico 1, que el 47% de las tareas fueron clasificadas por unanimidad por todos los usuarios con el criterio de fácil,

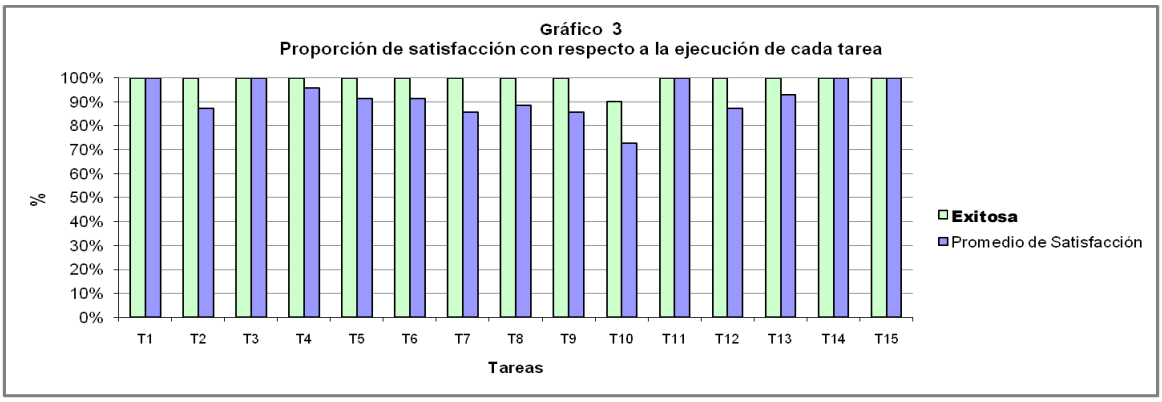
y de las restantes, el 88% fueron evaluadas con el mismo criterio por el 70% de los usuarios. Por otra parte resalta el criterio sobre la tarea T10, cuya evaluación fue definida por el 50% de los usuarios como de mediana dificultad, el 40% la calificó de difícil y un 10% necesitó asistencia para llevarla a cabo.



También se realizó un análisis de la proporción de las tareas ejecutadas satisfactoriamente con respecto a las que los usuarios no pudieron llevar a cabo por si solos. En el Gráfico 2 se evidencia lo anterior, realizándose el ajuste estadístico (margen de error basado en el rendimiento total de los usuarios) correspondiente a el hecho de que en el estudio solo participan una pequeña muestra de los especialistas que pueden utilizar el sistema. Como se observa, el 100% de tareas pueden ser ejecutadas de forma independiente como mínimo por el 80% de los usuarios, y teniendo en cuenta el intervalo de confianza, podría variar hasta un límite inferior del 70% sin incluir el caso crítico de la T10, cuyo límite inferior de ejecución correcta para todos los usuarios se expresa en un 60%.

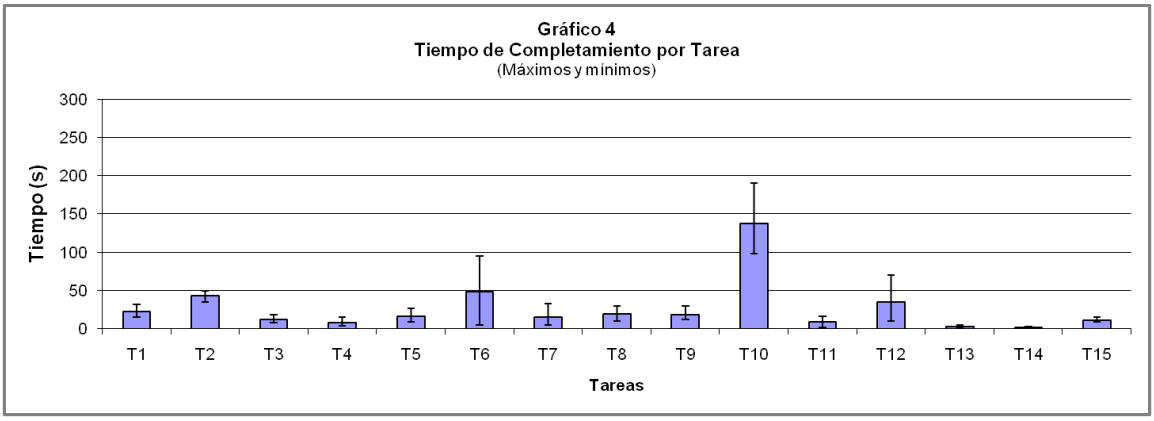


Con respecto al análisis correspondiente entre el nivel de satisfacción del usuario y la ejecución satisfactoria de cada tarea, mostrado en el Gráfico 3, se evidencia que en todos los casos, el nivel de ejecución satisfactoria de una tarea es igual o superior a la escala de conformidad que establece el usuario al respecto.



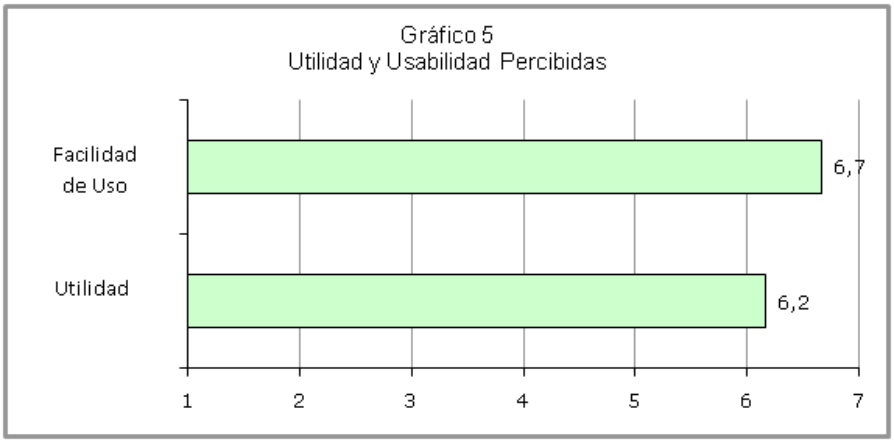
En la medición de los tiempos para llevar a cabo cada tarea por parte de los usuarios seleccionados, es importante señalar que existen tareas cuyos registros pueden ser bien variables, como son las T2 y T12, ya que dependen de la edición de datos en correspondencia con el estudio que se esté realizando y las exigencias de cada especialista en particular, y otras como la T6, cuyo tiempo de ejecución propio depende del método de evaluación (evaluación cruzada distribuida o no, u otro tipo). En el experimento, para el caso crítico se utilizó la base de datos de Promoters de la E. Colic, propuesta para validación de los algoritmos ByNet, BayesChaid, y BayesPSO por parte de sus desarrolladores [8], con 106 casos y 58 atributos, y se realizó la evaluación cruzada local en una PC de Core2 Duo E4400, a 2.0Ghz con 1Gb de RAM.

El resto de las pruebas fueron realizadas utilizando las bases de datos de la UCIML, alternando con las diferentes formas de evaluación. La T10 se ejecutó en un rango de 10 a 20 experimentos con 6 algoritmos para la prueba de Friedman. A continuación se muestran los tiempos promedios en el Gráfico 4 incluyendo los tiempos máximos y mínimos alcanzados para cada tarea.



Como se evidencia, de forma general los tiempos que tardaron los usuarios para realizar cada tarea no exceden de los 200 segundos.

Finalmente se muestra en el Grafico 5 el grado Utilidad del sistema, con un nivel de 6.2, y el grado de Usabilidad del mismo con un valor de 6.7, siendo ambos valores un promedio de las respuestas a las preguntas del cuestionario TAM realizadas a todos los usuarios. Estos resultados, sumados al análisis de las gráficas anteriores, demuestran la contribución del sistema con respecto a la disminución de la dificultad de uso de los clasificadores bayesianos por parte de los especialistas.



4. Guía de uso de la propuesta de solución

La aplicación gestiona los diferentes estudios como *experimentos*. Cada experimento se relaciona directamente con un conjunto de datos, por lo tanto el primer paso al crear un Experimento (Menú Archivo/ Nuevo) es seleccionar y editar los datos correspondientes al estudio.

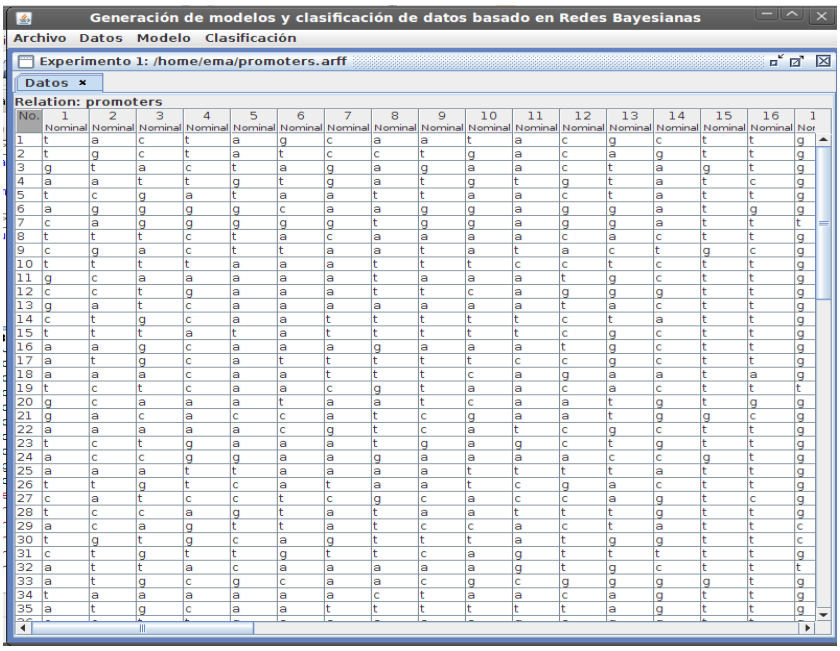


Fig.11 Vista del Panel de datos. Fichero promoters.arff.

El conjunto de datos puede cambiarse (Menú Datos/Cargar datos o desde la pestaña de selección de datos de Modelo/Nuevo escenario) y permanecerán mostrándose en la

Pestaña Datos (ver Fig. 11) mientras el usuario lo determine conveniente, de forma que puedan estar accesibles mientras se evalúan los clasificadores generados a partir de los mismos. Con los datos se pueden realizar varias operaciones como la inserción o eliminación de instancias, eliminación de atributos, reemplazo de valores, etc., antes de ser procesados.

Para la generación de nuevos modelos, es posible crear uno o tantos *escenarios* como el especialista estime conveniente (Menú Modelo/ Nuevo escenario). En cada escenario, en el panel desplegable derecho (Panel de Opciones) se definen las opciones de evaluación, se registran y seleccionan los parámetros de comparación que luego podrán ser vistos en la Pestaña de Comparación, se selecciona el algoritmo de aprendizaje estructural a utilizar (ByNet y BayesChaid en el paquete **`search/determinitic`**, y BayesPSO en el paquete **`search/global`**) y se definen los parámetros del mismo que aparecen automáticamente en la interfaz visual. Para generar el nuevo clasificador a partir de los datos de entrada y la especificación de las opciones anteriores, se debe oprimir el botón Generar (ver Fig.12).

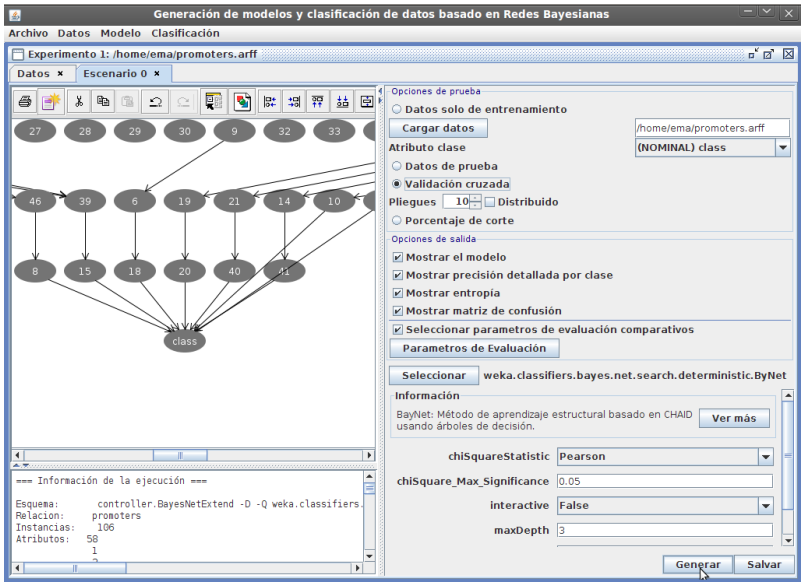


Fig.12 Vista del Pestaña Escenario. Clasificador bayesiano construido utilizando el algoritmo ByNet.

La red bayesiana generada aparecerá en el panel izquierdo desplegable central (Panel de Visualización), y el resumen de la generación y evaluación del clasificador se mostrará, según las opciones especificadas, en el panel izquierdo desplegable inferior (Panel Resumen).

El Panel de Visualización permite salvar la red en formato XML BIF [32], aplicar diferentes plantillas de organización de los nodos, mostrar un resumen de las probabilidades marginales de los nodos, así como los conglomerados, entre otras opciones. Además, seleccionando un nodo específico se podrá ver su tabla de probabilidad condicional.

Finalmente el modelo podrá ser guardado en la memoria física de la computadora oprimiendo el botón Salvar del panel de opciones.

Para comparar modelos se debe ir a la Pestaña de Comparación (Menú Modelo/Comparación) (ver Fig.13). En la misma se visualizan los parámetros de evaluación registrados y seleccionados para cada corrida de los diferentes algoritmos y escenarios que se ejecutaron en el sistema. Estos parámetros se presentan en forma de tabla, cuyos valores pueden ser organizados ascendente o descendentemente haciendo clic sobre el titulo de la columna. Además permite reorganizar las columnas haciendo clic sobre las mismas y arrastrándola hacia la posición deseada. Para el mejor análisis de los parámetros de evaluación es posible realizar gráficos de tipo barra, lineal y pastel, seleccionando los parámetros y las corridas a contrastar y visualizar en los mismos. Por otra parte, existe la opción de aplicar la prueba de Friedman accediendo a la ventana correspondiente (Menú Modelo/Prueba de Friedman,) y partiendo de la selección de varios experimentos y el parámetro de evaluación cuyos valores se quieren comparar, ejecutar la prueba.

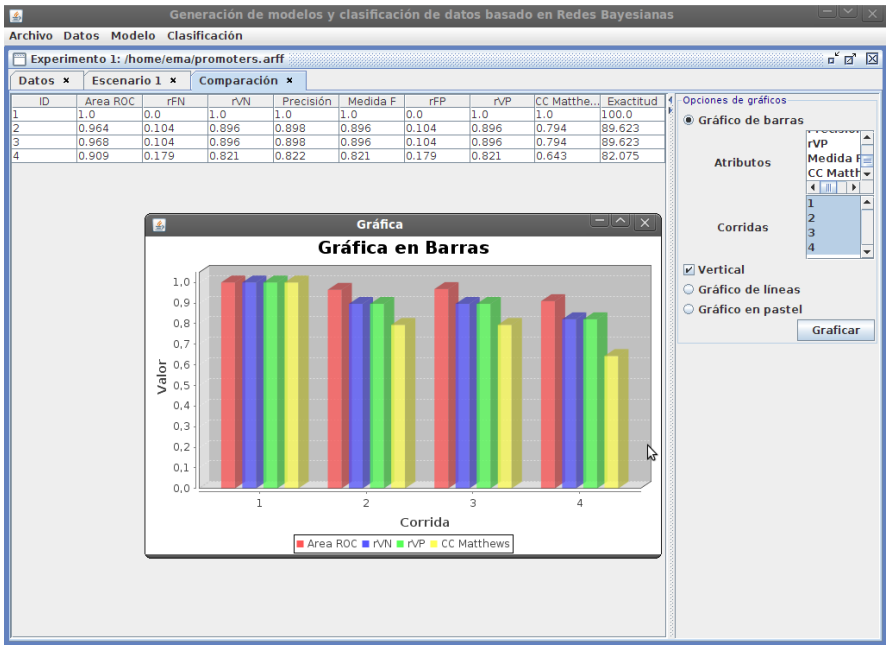


Fig.13 Vista del Panel Comparar. Comparación de los parámetros registrados por la ejecución de los algoritmos ByNet, BayesChaid, TAN y K2.

Para realizar la clasificación sobre un modelo anteriormente generado por la aplicación o a partir de uno guardado previamente, se debe ir a la Pestaña Clasificar (Menú Clasificación/Clasificar). En la misma se muestra un Panel de Opciones donde es posible seleccionar el modelo de clasificación a utilizar, el atributo clase, así como desde donde se obtendrán los datos a clasificar y si van a ser editados o no. Si los datos van a ser editados se muestra el Panel de Datos sin clasificar en el cual se podrán editar los datos de entrada para el clasificador. Es importante destacar que si un atributo es seleccionado como atributo clase, no podrá ser editado.

Una vez seleccionados y/o editados los datos, se procede a la clasificación pulsando el botón Clasificar. Aparecerá entonces el Panel de Datos Clasificados, donde se mostrarán, para cada instancia, los valores definidos anteriormente y el valor asignado al atributo clase, dando la posibilidad de guardar los datos clasificados de forma que puedan ser contrastados y/o utilizados en otros experimentos.

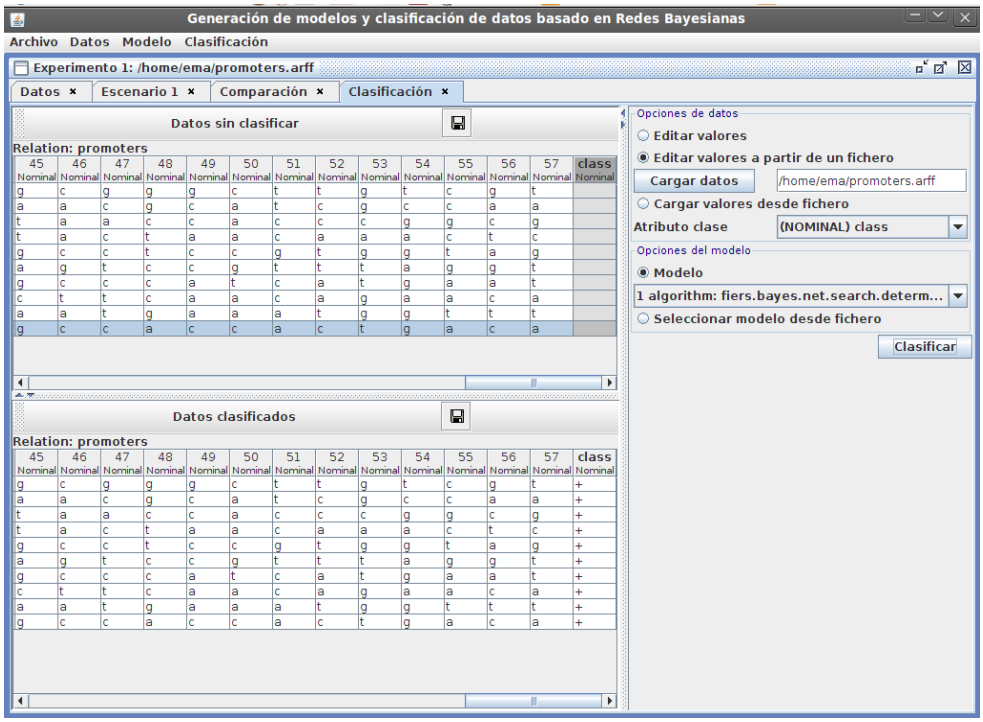


Fig.14 Vista de la Pestaña Clasificar. Clasificación de instancias utilizando el modelo aprendido con el algoritmo ByNet y la base de datos promoters.arff.

CONCLUSIONES

En el capítulo, se llevó a cabo la validación del sistema mediante diferentes tipos de pruebas, entre ellas las de unidad e integración. Las mismas posibilitaron mostrar dificultades del sistema y corregirlas oportunamente. Además se realizaron pruebas de usabilidad, que permitieron comprobar el grado de satisfacción del usuario respecto al uso y utilidad del sistema, en base a la realización de quince tareas y doce preguntas definidas por TAM, así como la cuantificación y análisis de tres métricas. Las pruebas de usabilidad arrojaron valores aproximados de 6.7 de un máximo de la escala de 7, lo cual es altamente representativo en cuanto a la disminución de la dificultad del uso de este tipo de modelo estadístico por parte de los especialistas.

CONCLUSIONES

En la presente investigación, a partir del estudio de los principales conceptos desde el punto de vista matemático que definen los procesos de generación, evaluación y uso de los clasificadores bayesianos, y la orientación del desarrollo hacia la extensión e integración de las funcionalidades de Weka, guiada por el Modelo de Integración Funcional, se diseñó e implementó una aplicación informática basada en tecnología libre que integró los tres procesos mencionados anteriormente, incluyendo los algoritmos de aprendizaje estructural ByNet, BayesChaid, y BayesPSO, optimizados para dominios Bioinformáticos y Médicos.

La aplicación fue concebida bajo el patrón arquitectónico en capas, de forma que el sistema pueda aprovechar las ventajas del mismo, y se realizaron diferentes tipos de pruebas que permitieron la validación del funcionamiento correcto del sistema. Las pruebas de usabilidad arrojaron valores aproximados de 6.7 de un máximo de la escala de 7, lo cual es altamente representativo en cuanto a la disminución de la dificultad del uso de este tipo de modelo estadístico por parte de los especialistas.

RECOMENDACIONES

- Realizar la implementación de una funcionalidad que permita seleccionar los posibles algoritmos de aprendizaje estructural a combinar en la prueba de Friedman por parte de los especialistas, mejorando así la usabilidad de esta función.

- Incluir otros algoritmos de propagación de evidencia para la clasificación de las nuevas instancias.

- Extender la utilización del sistema para la generación, evaluación y uso de clasificadores bayesianos a otros dominios de investigación, dado que el mismo además de presentar los algoritmos definidos para dominios bioinformáticos y médicos, incorpora los algoritmos de aprendizaje estructural tradicionales presentes en la plataforma Weka.

REFERENCIAS BIBLIOGRÁFICAS Y BIBLIOFRAFÍA

1. **Elvira, Proyecto.** [En línea] http://www.ia.uned.es/~elvira/.
2. HUGIN EXPERTS A/S. [En línea] http://www.hugin.com.
3. **Norsys.** Norsys. [En línea] www.norsys.com..
4. **MIC. Cuba.** Directivas generales y políticas. [En línea] Junio de 2011. http://www.cubagob.cu/ingles/des_eco/mic/mic_directivas/directivas.htm.
5. **The University of Waikato.** Weka. [En línea] enero de 2011. www.cs.waikato.ac.nz/ml/weka/.
6. **Cozman, Fabio Gagliardi.** JavaBayes, Bayesian Networks in Java. User manual and download. [En línea] 2001.
7. **Siegel, S.** *Diseño Experimental no paramétrico.* s.l. : Edic. Rev.: 346 páginas, 1987.
8. **Chávez Cárdenas, María del C., Grau Á., Ricardo y Casas, Gladys.** *Modelos de Redes Bayesianas en el estudio de secuencias genómicas y otros problemas biomédicos.* Santa Clara, Villa Clara, Cuba : Universidad Central "Marta Abreu" De Las Villas, Facultad de Matemática, Física y Computación, 2008.
9. **Matthews, B.W.** *Comparison of the predicted and observed secondary structure of T4 phage lysozyme.* s.l. : Biochim. Biophys., 1975.
10. **Duda, R., y Hart P.** Pattern Classification and Scene Analysis. 1973.
11. **Bayes, Thomas.** *An Essay towards solving a Problem in the Doctrine of Chances.* Philosophical Transactions of the Royal Society of London : s.n., 1763.
12. **Pearl, J.** *Probabilistic reasoning in intelligent systems: networks of plausible inference.* San Mateo, California : s.n., 1988.
13. **Schulz, Sr. Gastón.** *Un ambiente integrado de clasificación, selección y ponderación de reglas basado en sistemas inteligentes.* s.l. : Universidad de Buenos Aires.
14. **autores, Grupo.** *Redes Bayesianas. Capítulo 3. Redes bayesianas y su propagación.*
15. **Sucar, Luis Enrique.** Redes Bayesianas. Sta. María Tonantzintla, Puebla, 72840, México : s.n.
16. **Chow, C. and Liu.** Approximating discrete probability distribution with dependence trees. 1968., Vols. 114: 462- 467.
17. **Rebane, G. y Pearl, J.** *The recovery of causal poly- trees from statistical data.* s.l. : Int. J. Approxi. Reasoning, 1988.
18. **Cooper, G. F. y Herskovits, E. H.** A Bayesian methods for the induction of probabilistic networks from data. 1992.
19. **Buntine, W. L.** Operation for Learning with Graphical Models. 1994.
20. **Spirtes, P. y Meek, C.** *Learning Bayesian networks with discrete variables from data.* s.l. : In Proceeding of the First International Conference on Knowledge Discovery and Data Mining: 294- 299, 1995.
21. **Saucier, R.** *Computer Generation of Statistical Distributions. Report of Army Research Laboratory paper.* 2000.
22. **Neapolitan, R. E.** *Probabilistic Reasoning in Expert Systems: Theory and Algorithms Wiley-Interscience.* New York : s.n., 1990 .

23. **Puga, Jorge López, y colectivo de autores.** *Las Redes Bayesianas como herramientas de modelado en psicología. Anales de Psicología.* Murcia, España : Universidad de Murcia, diciembre, 2007.
24. **Silva, L. C. y Muñoz, A.** *Debate sobre métodos frecuentistas vs bayesianos.* s.l. : Gaceta Sanitaria 14(6): 482-494, 2000.
25. **Spirtes, P., Glymour, C., y Scheines, R.** *Causation, prediction and search (2ª ed.).* s.l. : Cambridge, MA: MIT Press, 2000.
26. **Fawcett, Tom.** ROC Graphs: Notes and Practical Considerations for Researchers. HP Laboratories. March 16, 2004.
27. **Kohavi, Ron.** *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.* Stanford University : s.n.
28. **Madsen, A. L., Jensen, F., Kjærulff, U. y Lang, M.** *The HUGIN tool for probabilistic graphical models. International Journal of Artifcial Intelligence Tools.* (2005).
29. **Santander, Nuria Alonso.** *Sistema de ayuda a la decisión para cirugía de cataratas. Memoria de Tesis Doctoral.* Madrid : Universidad Autónoma de Madrid, Facultadad de Medicina , 2010.
30. **Murphy, Kevin.** Software Packages for Graphical Models / Bayesian Networks. [En línea] Last Update 2010. http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html.
31. **Chávez Cárdenas, María del Carmen y otros.** Red Bayesiana a partir de factores de riesgo de la Hipertensión Arterial. 2007.
32. **The University of Waikato.** WEKA Manual for Version 3-6-2. s.l. : The University of Waikato, 2010.
33. **Linthicum, D.S.** *Enterprise Application Integration.* Boston-USA. : Addison-Wesley, 2000.
34. **Frantz, Rafael Z.** *Integración de aplicaciones. Un lenguaje específico de dominio.* Universidad de Sevilla : s.n., Junio, 2008.
35. **Palacio, David Melend, Rodríguez, Abel Rionda.** *Tipos de Integración .* Universidad de Oviedo, Asturias. España : s.n.
36. **Montilva, Jonás A. y Chacón R, Edgar.** *Automatización de Sistemas e Integración de Software.* Mérida, Venezuela : Universidad de Los Andes.
37. **Tse, William.** *Enterprise Application Integration.* s.l. : UCL Computer Science.
38. **Bertie, Andrew James.** Java Statistical Classes. [En línea] 2005. http://www.jsc.nildram.co.uk/.
39. **Davis, F. D.** Perceived usefulness, perceived ease of use, and user acceptance of information technology. 1989.
40. **Wuensch, Karl L.** *What is a Likert Scale? and How Do You Pronounce 'Likert?* East Carolina University : s.n., 2005.
41. **Molina L., José Manuel y García H., Jesús.** *Técnicas de análisis de datos.Aplicaciones prácticas utilizando Microsoft Excel y Weka.* Madrid. España : Universidad Carlos III de Madrid, 2006.
42. **Sheskin, D.J.** *Handbook of Parametric and Nonparametric Statistical Procedures. .* s.l. : CRC Press, 2003.

43.**Berzal, Fernando.** *Métodos de clasificación.* Granada. Spain : Deparment of Computer Science and Artificial Intelligence, 2008.
44. **Díez, F.J.** Aplicaciones de los modelos gráficos probabilistas en medicina. 1998.
45. **Marcucci A., Gauthier.** *Implementación y evaluación de métodos de clasificación: Estadísticos, de agrupamiento, árboles de decisión y metaheurísticos.* Mayo de 2005.
46. **Galán, Severino Fernández.** *Redes bayesianas temporales: aplicaciones médicas e industriales.* Madrid, España : Universidad Nacional de Educación a Distancia, 2002.
47. **García, Alberto Muñoz.** Clasificación de datos multivariantes: análisis discriminante. *OpenCourseWare - Universidad Carlos III de Madrid.* [En línea] 03 de 03 de 2008. [Citado el: 10 de 02 de 2011.] http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/.
48. **Mandeville, Peter B.** *Tips Bioestadísticos, Evaluación del desempeño.* s.l. : Ciencia UANL / VOL. XII, No. 2, abrilL 2009.
49. **Puga, Jorge López y García, Juan García.** *Sistemas de Tutorización Inteligente Basados en Redes Bayesianas. Revista Electrónica de Metodología Aplicada.* Almería, España. : Universidad de Almería, 2008.
50. **Weiss, S.M y Indurkhya.** *Predictive Data Mining.* San Francisco : Morgan Kaufmann, 1998.
51. **Beltrán, D. Francisco Roche.** *Métodos para obtener conocimiento utilizando Redes Bayesianas y procesos de aprendizaje con algoritmos evolutivos. Memoria de Tesis Doctoral.* Sevilla : Universidad de Sevilla, Departamento de Lenguajes y Sistemas Informáticos, Septiembre 2002.
52. **López, Joseba Esteban y Dolado, José Javier.** *Estudio de los métodos de estimación: AHP y redes Bayesianas*. Universidad del País Vasco U.P.V./E.H.U : Departamento de Lenguajes y Sistemas.
53. **Villanueva, David A. Velasco.** *Redes Bayesianas. Inteligencia Artificial II.*

ANEXOS

Anexo 1. Filtro de las aplicaciones para Redes Bayesianas de Murphy [30] basado en los parámetros especificados en el Capítulo1, Sección 4.

Name	Authors	Src	API	Exec	GUI	Params	Struct	Free	Inference
Banjo	**Hartemink**	**Java**	**Y**	**W,U,M**	**N**	**N**	**Y**	**0**	**none**
Bayda	U. Helsinki	Java	Y	WUM	Y	Y	N	0	?
BNT	**Murphy (U.C.Berkeley)**	**Matlab/ C**	**Y**	**WUM**	**N**	**Y**	**Y**	**0**	**Many**
Vibes	Winn & Bishop (U. Cambridge)	Java	Y	WU	Y	Y	N	0	Variational

Tabla 2. Intersección entre el conjunto de aplicaciones que realizan aprendizaje paramétrico (Params = Y) (sombreado) y las que realizan aprendizaje estructural (Struct = Y) (letras verdes) de la red bayesiana.

Anexo 2. Cuestionario de percepción de utilidad y facilidad de uso (TAM) [39]

	Cuestionario de percepción de utilidad y facilidad de uso	Media	Fuerte discrepancia … Fuerte concordancia **1** 2 3 4 5 6 **7**
1	El uso de este producto me permite realizar tareas con mayor rapidez.	6.9	
2	El uso de este producto mejora mi rendimiento actual.	6.0	
3	El uso de este producto aumenta mi productividad.	5.8	
4	El uso de este producto me hace más eficaz.	5.5	
5	El uso de este producto hace más fácil realizar mi trabajo.	6.0	
6	Me parece útil del producto.	6.8	
7	Aprender a utilizar este producto fue fácil para mí.	6.3	
8	Con este producto es fácil hacer lo que yo lo quiero que haga.	6.9	

9	Mi interacción con este producto es de una forma clara y comprensible.	6.5	
10	Este producto es flexible para interactuar con el.	7.0	
11	Fue fácil para mí volverme hábil en el uso de este sistema.	6.3	
12	Encontré el sistema fácil usar.	7.0	
	UTILIDAD	6.2	
	FACILIDAD DE USO	6.7	

Tabla 3. Resultados del cuestionario de percepción de utilidad y facilidad de uso (TAM).

GLOSARIO

AB: Asma Bronquial

API: Application Program Interface

CIGB: Centro de Ingeniería Genética y Biotecnología

CIM: Centro de Inmunología Molecular

DM: Diabetes Mellitus

GUI: Graphical User Interface

HTA: Hipertensión Arterial

RB: Red Bayesiana

RF: Requisito Funcional

RNF. Requisito No Funcional

SO: Sistema Operativo

T-Arenal: Plataforma potente de cálculo distribuido desarrollada por el Grupo de Bioinformática de la Universidad de Ciencias Informáticas.

TPC: Tabla de Probabilidad Condicional

UCI: Universidad de Ciencias Informáticas

UCIML: Bases de datos del repositorio de aprendizaje automático, del inglés UCI Repository of Machine-Learning Databases

UCLV: Universidad Central “Marta Abreu” de Las Villas