Título: Sistema multiagente para la extracción de conocimiento en bases de datos.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Jaime Alan Gutiérrez Cruz

Yanelis Cedeño Bastida

Tutor: MSc. Yuniesky Coca Bergolla

Co-Tutor: Ing. Leduan Bárbaro Rosell Acosta

La Habana, Cuba

Junio de 2017

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

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

_____
Jaime Alan Gutiérrez Cruz
Firma del Autor

_____
Yanelis Cedeño Bastida
Firma del Autor

_____
MSc. Yuniesky Coca Bergolla
Firma del Tutor

_____
Ing. Leduan Bárbaro Rosell Acosta
Firma del Co-Tutor

DEDICATORIA

Este trabajo y todos sus resultados se los dedico a mi hermana, gracias por el amor, la entrega, la paciencia y la confianza que siempre me has dedicado. Nunca podré llegar a agradecerte los suficiente por todos los sacrificios. Gracias por creer en mi soñar conmigo esta carrera. Si hoy estoy aquí es por ti. Te adoro.

Yanelis

Este trabajo y todos sus resultados se los dedico a mis padres que me han dado todo su apoyo y de los que me siento orgulloso, y a mis abuelas que han sido más que abuelas para mí.

Jaime

AGRADECIMIENTOS

Le agradezco a mi hermana por todo el amor y la dedicación.

A mi mamá, mis abuelos Sonia y Ernesto de una forma u otra me ayudaron a convertirme en la persona que soy hoy.

A nuestro tutor, eres un ser humano extraordinario, gracias por dedicarnos tanto tiempo, por confiar en nosotros, sin ti esto no hubiera sido posible

A nuestro co-tutor gracias por el apoyo y la atención

A mi compañero de tesis y su familia, gracias por el cariño y la preocupación.

A Yolexis gracias por estar ahí, siempre tendrás un lugarcito en mi corazón.

A mis compañeros de Universidad, gracias por los buenos momentos.

A Gloria, Lamis, Armando, Yasiel y David, conocerlos ha sido una experiencia única, espero que nos sigamos viendo.

A los profesores que, a través de este largo camino, han sabido guiarme y que son estupendos profesionales.

A Diana estoy feliz por haberte conocido.

A las personas que ya no están entre nosotros pero que siempre me dieron su cariño y lucharon para que cumpliera mi sueño.

A la UCI, y en especial a la facultad 5 por hacer realidad mi sueño y el de muchos otros.

Yanelis

AGRADECIMIENTOS

A mis padres que me han apoyado siempre en todas las decisiones que he tomado y me han enseñado a ser quien soy.

A mi hermana que, a pesar de las discusiones de hermanos, nos queremos y apoyamos de una forma que solo nosotros entendemos.

A mis abuelos Amada, Andrea y Alberto, que me han enseñado el valor de la familia.

A mi tío José Luis que ha sido como un segundo padre, y mi tía Regla que nos quiere a mi hermana y a mí como sus hijos.

A Rafael, ese amigo que nunca olvida y siempre está, y aunque no compartamos la sangre somos hermanos.

A Yanelis, sin la que no hubiese podido terminar este trabajo y más que una compañera en el mismo ha sido una gran amiga en estos cinco años, no me olvido tampoco de su hermana que nos ha apoyado mucho.

A Gloria, amiga que me ha ayudado más de lo que piensa, si hoy soy mejor persona, parte es gracias a ella.

A José Manuel, Gabriel y David que mejores amigos que estos con los que he disfrutado mucho; y no me olvido de aquellos amigos del grupo 4: Randy, Damian, Yasser, Alejandro, etc.

A Yaima, la única persona que me conoce bien y me saca de mis cabales con el objetivo de hacerme olvidar los problemas, ella casi nunca está triste a pesar de lo mal que este.

A nuestro tutor, Yuniesky, si hoy estamos aquí es gracias a él que siempre nos dio su apoyo y confianza, mejor tutor no podríamos haber tenido.

A nuestro co-tutor, Leduan, por su ayuda y apoyo.

A los profesores y personas que han intervenido en mi formación académica, es gracias a ustedes que me convierto en un buen profesional.

A todos los que he conocido durante todos estos años que de una u otra forma han dejado huella en mi vida.

Jaime

RESUMEN

El Entorno para el Análisis del Conocimiento de la Universidad de Waikato (Weka) es una herramienta destinada a la aplicación de la Minería de Datos. Esta brinda la opción de aplicar varios algoritmos de clasificación a un determinado problema, sin embargo, carece de la realización de un análisis automático de la comparación de los resultados. Además, los datos deben encontrarse en el formato Attribute-Relation File Format (ARFF), lo cual dificulta la aplicación de sus algoritmos a bases de datos en otros formatos. El continuo avance de la Inteligencia Artificial ha posibilitado el surgimiento de los sistemas multiagentes, los cuales, integrados con la Minería de Datos, pueden ser beneficiosos para el perfeccionamiento de los resultados en ambas esferas. Esta investigación persigue como objetivo desarrollar un sistema multiagente para determinar el mejor algoritmo de los ofrecidos por Weka, mediante el cálculo de un conjunto de métricas para analizar los algoritmos de clasificación y agrupamiento. Dentro de sus funciones cuenta con un proceso de transformación de bases de datos de PostgreSQL a archivos ARFF. El sistema propuesto fue objeto de pruebas con diversas bases de datos, se obtuvieron los resultados esperados, tanto en rendimiento como en nivel de cumplimiento de los requisitos trazados, lo cual demuestra el cumplimiento de los objetivos iniciales.

Palabras claves: Agrupamiento, Clasificación, Minería de Datos, Sistemas Multiagentes, Weka.

Contenido

Índice de tablas

Índice de figuras

INTRODUCCIÓN

A través de los años, el ser humano ha tratado de entender cómo es capaz de pensar. El campo de la Inteligencia Artificial (IA) no solo intenta comprender, sino que también se esfuerza en construir entidades inteligentes. La IA es una de las ciencias más recientes y abarca una gran variedad de sub-campos, que van desde áreas de propósito general, como el aprendizaje y la percepción, a otras más específicas como el ajedrez, la demostración de teoremas matemáticos y el diagnóstico de enfermedades, entre otras (1).

Uno de los ejes fundamentales de la IA son los sistemas multiagentes (SMA). Un sistema multiagente permite la convivencia de varios agentes inteligentes que interactúan entre sí para darle solución a objetivos específicos, cada uno de los agentes debe ser capaz de atender tareas globales y específicas. Un sistema multiagente se enfrenta al reto de mejorar la capacidad de aprendizaje de los agentes (2) y tienen aplicaciones dentro de la industria, la robótica, la salud, la educación y el entretenimiento, como es el caso de los videojuegos. En el mundo se han desarrollado diversos SMA encaminados a solucionar problemas de la vida cotidiana, entre los que se encuentran ALZheimer Multi-Agent System (ALZ-MAS), el Sistema Inteligente de Monitoreo en Anestesiología, LATIDO e Intelligent System for Breast Cancer (3). En Cuba no se han elaborado muchos sistemas que sigan el paradigma orientado a agentes. Uno de los más reconocidos es el Sistema de Filtrado de Paquetes por Contenido (FILPACON), el cual cuenta con el Motor de Clasificación Inteligente de Contenidos (MOCIC) que constituye un ejemplo del uso de sistemas inteligentes. Este sistema fue desarrollado por la Universidad de las Ciencias Informáticas (UCI) (3).

Un área donde se han obtenido resultados con los SMA es la Minería de Datos (MD). La MD brinda apoyo al trabajo con grandes volúmenes de datos, permitiendo la extracción de información relevante y dando soporte en la toma de decisiones, por lo que se ha vinculado con áreas como finanzas, análisis de mercado, medicina, telecomunicaciones, entre otras. Sin embargo, se encuentra ante la paradoja del continuo crecimiento y heterogeneidad de los datos.

El Entorno para el Análisis del Conocimiento de la Universidad de Waikato (Weka, por sus siglas en inglés) es una herramienta que posibilita la aplicación de la Minería de Datos y facilita la realización de técnicas de clasificación y agrupamiento. Sin embargo, seleccionar cuál algoritmo utilizar para resolver un problema concreto puede ser engorroso, pues hay que aplicarle varios de los algoritmos para verificar cuál arroja las mejores soluciones. Además, se deben tener las bases de datos en el formato Attribute-Relation File Format

(ARFF) de Weka para poder aplicarlos y a pesar de permitir el trabajo con otros tipos de bases de datos presenta no conformidades en la conexión con PostgreSQL, en ocasiones con el driver y otras no encuentra sus clases.

Por lo planteado anteriormente surge la necesidad de dar solución al siguiente **problema de investigación**:

¿Cómo identificar de forma automática la mejor solución brindada por los algoritmos de Weka para resolver un problema de Minería de Datos?

Para dar solución al problema se define como **objetivo general** de la investigación: Desarrollar un sistema multiagente que determine el mejor algoritmo de clasificación o agrupamiento a aplicar sobre una base de datos mediante el cálculo de métricas de evaluación.

Se define como **objeto de estudio** el proceso de Minería de Datos y como **campo de acción** los Sistemas para clasificación y agrupamiento en bases de datos.

Se definieron las siguientes tareas de investigación para alcanzar el resultado deseado:

- Análisis de la integración de las herramientas Java Agent Development Framework (JADE) y Weka.
- Integración de JADE y Weka.
- Síntesis de una búsqueda histórico-lógica de los elementos fundamentales sobre los sistemas multiagentes desarrollados en JADE, los algoritmos de clasificación y de agrupamiento.
- Selección de las herramientas y metodología a utilizar.
- Análisis y diseño del sistema, siguiendo la metodología seleccionada y documentación de dichos procesos.
- Implementación del sistema.
- Validación del sistema.

Como parte del cumplimiento de las tareas definidas se emplearon los siguientes **métodos científicos**:

1- Teóricos:

- **Histórico-Lógico**: este método se utilizó para el análisis del estado del arte, se obtienen los aspectos más relevantes sobre los sistemas multiagentes, la Minería de Datos y los algoritmos de clasificación y agrupamiento.
- **Analítico-Sintético:** este método posibilitó la síntesis de la información recopilada durante el proceso de investigación.

- **Modelación**: a través de este método se modeló el prototipo funcional de la aplicación y su arquitectura.

2- Empíricos:

Estudio de la documentación: este método se utilizó para realizar la búsqueda exhaustiva de la información referente a los distintos temas tratados en el estudio del estado del arte.

- **Test o prueba**: se utilizó para evaluar la calidad de los resultados alcanzados por los algoritmos de clasificación y agrupamiento.
- **Experimentación**: se experimentó con varios algoritmos de clasificación y agrupamiento para determinar cuál arrojaba mejores resultados.

El presente documento está estructurado en tres capítulos. En el Capítulo 1 se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. En el Capítulo 2 se expone una propuesta de solución basada en SMA. En el Capítulo 3 se muestran los resultados de la propuesta desarrollada y se exponen las pruebas realizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo aborda la información recopilada sobre la Minería de Datos, se definen y explican las técnicas de Minería de Datos y se caracterizan los algoritmos de clasificación y agrupamiento. Además, se hace un breve estudio de los agentes inteligentes y SMA, así como una descripción de las herramientas y metodología que se emplearán.

1.1 Minería de Datos

En la actualidad la cantidad de datos que ha sido almacenada en las bases de datos excede la habilidad humana para reducirlos y analizarlos sin el uso de técnicas de análisis automatizadas. Muchas bases de datos comerciales transaccionales y científicas crecen a una proporción desmesurada.

El Descubrimiento de Conocimiento en Bases de Datos (KDD, por sus siglas en inglés), atendiendo al criterio de García Herrero y Molina López en (4), es el proceso completo de extracción de información que se encarga de la preparación de los datos y de la interpretación de los resultados obtenidos. También se define como el proceso no trivial de identificación, en los datos, de patrones válidos, nuevos, potencialmente útiles y finalmente comprensibles. Tareas comunes en dicho proceso son la inducción de reglas, los problemas de clasificación y agrupamiento, el reconocimiento de patrones, el modelado predictivo, la detección de dependencias, entre otras (4).

El proceso de KDD se inicia con la identificación de los datos. Para ello hay que conocer qué datos se necesitan, dónde se pueden encontrar y cómo conseguirlos. Una vez que se dispone de dichos datos, se deben seleccionar aquellos que sean útiles para los objetivos propuestos. Posteriormente se preparan y se procede a la Minería de Datos, proceso en el que se seleccionan las herramientas y técnicas adecuadas para lograr los objetivos pretendidos. Tras este proceso se realiza el análisis de los resultados, con lo que se obtiene el conocimiento pretendido.

Minería de Datos es un término genérico que engloba resultados de investigación, técnicas y herramientas usadas para extraer información útil de grandes bases de datos. La Minería de Datos es una parte del proceso completo de KDD, sin embargo, en buena parte de la literatura los términos MD y Descubrimiento de Conocimiento en Bases de Datos se identifican como si fueran lo mismo. Concretamente, el término MD es usado comúnmente por los estadísticos, analistas de datos y por la comunidad de administradores de sistemas informáticos como todo el proceso del descubrimiento, mientras que el término KDD es utilizado más por los especialistas en IA (4).

1.2 Técnicas de Minería de Datos

Las técnicas para la extracción de información relevante que se utilizan como parte del proceso de extracción de conocimiento y a las cuales se hizo referencia en el epígrafe 1.1, desempeñan un papel preponderante dentro de la Minería de Datos. Su principal objetivo es obtener patrones o modelos a partir de los datos recopilados. Decidir si los modelos obtenidos son útiles o no, suele requerir una valoración subjetiva por parte del usuario. Dichas técnicas se clasifican en dos grandes categorías: supervisadas o predictivas y no supervisadas o descriptivas (5).

Dentro de las técnicas supervisadas están las de clasificación y dentro de estas se pueden mencionar:

- Tabla de Decisión.
- Árboles de Decisión.
- Inducción de Reglas.
- Bayesiana.
- Basado en Ejemplares.
- Redes de Neuronas.
- Lógica Difusa.
- Técnicas Genéticas.

Como parte de las técnicas no supervisadas se encuentran las de agrupamiento:

- Numérico.
- Conceptual.
- Probabilístico.

Varios autores coinciden en que una técnica constituye el enfoque conceptual para extraer la información de los datos, en general, es implementada por varios algoritmos. Cada algoritmo representa, en la práctica, la manera de desarrollar una determinada técnica gradualmente, de forma que es preciso entender los algoritmos para saber cuál es la técnica más apropiada para cada problema. Así mismo es preciso comprender los parámetros y las características de los algoritmos para preparar los datos a analizar (5).

El aprendizaje inductivo no supervisado estudia el aprendizaje sin la ayuda del *maestro*; es decir, se aborda el aprendizaje sin supervisión, que trata de ordenar los ejemplos en una jerarquía según las regularidades en la distribución de los pares *atributo-valor* sin la guía del atributo especial *clase*. Este es el proceder de los sistemas que realizan agrupamiento conceptual y de los que se dice también que adquieren nuevos conceptos.

En el aprendizaje inductivo supervisado existe un atributo especial, normalmente denominado *clase*, presente en todos los ejemplos, el cual especifica si el ejemplo pertenece o no a un cierto concepto, que es el objetivo del aprendizaje. Mediante una generalización del papel del atributo clase, cualquier atributo puede desempeñar ese papel, lo cual convierte la clasificación de los ejemplos según los valores del atributo en cuestión, en el objeto del aprendizaje. Es decir, el objetivo del aprendizaje supervisado es: a partir de un conjunto de ejemplos de entrenamiento, de un cierto dominio D de ellos, construir criterios para determinar el valor del atributo *clase* en un ejemplo cualquiera del dominio. Esos criterios están basados en los valores de uno o varios de los otros pares (*atributo-valor*) que intervienen en la definición de los ejemplos (5).

1.3 Algoritmos de clasificación

Dentro de las técnicas supervisadas de MD se encuentra la clasificación, proceso de dividir un conjunto de datos en grupos mutuamente excluyentes, de tal forma que cada miembro de un grupo esté lo más cerca posible de otros miembros del mismo grupo y lo más lejos posible de miembros de grupos diferentes, donde la distancia se mide con respecto a las variables especificadas que se quieren predecir (6).

Se han escogido cuatro de los algoritmos de clasificación más utilizados y representativos que implementa Weka (7). El K-Vecinos Más Próximos (KNN, por sus siglas en inglés) como método de clasificación directa, sin la creación de un modelo explícito; J48 dentro de los árboles de decisión; NaiveBayes como método probabilístico y Optimización Mínima Secuencial (SMO, por sus siglas en inglés) como algoritmo matemático. En la presente investigación se han tomado los algoritmos anteriores como los básicos pues brindan mejores resultados dependiendo del tipo de problema al que se enfrentan.

1.3.1 Algoritmo KNN

El aprendizaje basado en ejemplares o instancias tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros solo los más representativos y en otros los incorrectamente clasificados cuando se clasifican por primera vez. La clasificación posterior se realiza por medio de una función que mide la proximidad o parecido. Dado un ejemplo para clasificar se le clasifica de acuerdo al ejemplo o ejemplos más próximos (8).

El método KNN es una técnica de clasificación basada en ejemplares. Se denomina método porque es el armazón de un algoritmo que admite el intercambio de la función de proximidad

dando lugar a múltiples variantes. La función de proximidad puede decidir la clasificación de un nuevo ejemplo atendiendo a la clasificación del ejemplo o de la mayoría de los k ejemplos más cercanos. Admite también, funciones de proximidad que consideren el peso o coste de los atributos que intervienen, lo que permite, entre otras cosas, eliminar los atributos irrelevantes.

KNN permite que los atributos de los ejemplares sean simbólicos (nominal), fechas (date) y numéricos (numeric, real, integer), así como que haya atributos sin valor (o valores perdidos) (8).

1.3.2 Algoritmo J48

El aprendizaje de árboles de decisión está englobado como una metodología del aprendizaje supervisado y consiste en una representación del conocimiento relativamente simple y es una de las causas por la que los procedimientos utilizados en su aprendizaje son más sencillos que los de sistemas que utilizan lenguajes de representación más potentes, como las redes semánticas (8).

Un árbol de decisión puede interpretarse como una serie de reglas compactadas para su representación en forma de árbol. Dado un conjunto de ejemplos, estructurados como vectores de pares ordenados *atributo-valor*, de acuerdo con el formato general en el aprendizaje inductivo a partir de ejemplos, el concepto que estos sistemas adquieren durante el proceso de aprendizaje consiste en un árbol. Cada eje está etiquetado con un par *atributo-valor* y las hojas con una *clase*, de forma que la trayectoria que determinan desde la raíz los pares de un ejemplo de entrenamiento alcanzan una hoja etiquetada con la clase del ejemplo. La clasificación de un ejemplo nuevo del que se desconoce su clase se hace con la misma técnica, solamente que en ese caso al atributo *clase*, cuyo valor se desconoce, se le asigna de acuerdo con la etiqueta de la hoja a la que se accede con ese ejemplo (8).

El primer sistema que construyó árboles de decisión fue CLS de Hunt, desarrollado en 1959 por psicólogos, como un modelo del proceso cognitivo de formación de conceptos sencillos. En 1979 Quinlan desarrolla el sistema ID3, conceptualmente es fiel a la metodología de CLS, pero le aventaja en el método de expansión de los nodos. La versión final, presentada por su autor Quinlan como un sistema de aprendizaje, es el sistema C4.5 (8).

El algoritmo C4.5 es una extensión del ID3 pues este último presentaba dificultades a la hora de trabajar con los valores de atributos continuos. En (9) se expresa que C4.5 permite el uso del concepto razón de ganancia, la construcción de árboles de decisión cuando

algunos de los ejemplos presentan valores desconocidos para algunos de los atributos, la poda de dichos árboles, la obtención de Reglas de Clasificación y el trabajo con atributos que presenten valores continuos (8).

El algoritmo J48, que se pretende emplear en la presente investigación, es una implementación del C4.5, donde la generación de reglas de clasificación a partir del árbol de decisión no es posible y al que se le amplían algunas funcionalidades tales como:

- Los tipos de atributos admitidos pueden ser simbólicos, fechas y numéricos. Se permiten ejemplos con faltas en dichos atributos, tanto en el momento de entrenamiento como en la predicción de dicho ejemplo, pero la clase debe ser nominal.
- Se permiten ejemplos con peso.

1.3.3 Algoritmo NaiveBayes

Los clasificadores Bayesianos son clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de clase, como la probabilidad de que una muestra dada pertenezca a una clase particular. La clasificación Bayesiana se basa en el teorema de Bayes y los clasificadores Bayesianos han demostrado una alta exactitud y velocidad cuando se han aplicado a grandes bases de datos (10). Diferentes estudios comparando los algoritmos de clasificación han determinado que un clasificador Bayesiano sencillo conocido como el clasificador “NaiveBayes” es comparable en rendimiento a un árbol de decisión y a clasificadores de redes de neuronas (8).

El clasificador NaiveBayes se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos en un conjunto finito de clases. Clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos. Estos clasificadores asumen que el efecto de un valor del atributo en una clase dada, es independiente de los valores de los otros atributos. Esta suposición se llama independencia condicional de clase. Esta simplifica los cálculos involucrados y, en este sentido, es considerado "ingenuo” (*naive*). Esta asunción es una simplificación de la realidad. A pesar del nombre del clasificador y de la simplificación realizada, el NaiveBayes tiene un funcionamiento correcto, especialmente cuando se filtra el conjunto de atributos seleccionado para eliminar redundancia, con lo que se elimina también dependencia entre datos. Además, permite que los atributos de los ejemplares sean simbólicos y numéricos (8).

1.3.4 Algoritmo SMO

Entre las máquinas de aprendizaje más populares aplicadas a la clasificación están las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés), las cuales básicamente funcionan mapeando las distintas características de los elementos que se desean clasificar a un espacio vectorial, en el cual trazan un hiperplano que separa los vectores de una clase de los del resto. Las SVM son reconocidas por tener un funcionamiento relativamente sencillo, asociado a una serie de propiedades teóricas y prácticas bastante atractivas, además de que pueden ser modificadas con facilidad para abordar problemas con datos que no son linealmente separables o que contienen mucho ruido (11).

Se utiliza el método de Optimización Mínima Secuencial (SMO, por sus siglas en inglés) para entrenar el algoritmo SVM. SMO divide la gran cantidad de problemas de Programación Cuadrática (QP, por sus siglas en inglés) que necesitan ser resueltos en el algoritmo SVM por una serie de problemas QP más pequeños (12).

Se utiliza el SMO implementado por John Platt para entrenar el clasificador de Soporte Vectorial. De manera general esta implementación reemplaza todos los valores perdidos (o vacíos) y transforma los atributos nominales en binarios. También, normaliza todos los atributos por defecto. En ese caso los coeficientes en la salida están basados en los datos normalizados y no en los originales, esto es importante para la interpretación del clasificador. SMO permite atributos simbólicos y numéricos.

1.4 Algoritmos de Agrupamiento

El agrupamiento es una técnica no supervisada de MD, permite la identificación de tipologías o grupos donde los elementos guardan gran similitud entre sí y muchas diferencias con los de otros grupos.

Las herramientas de agrupamiento se basan en técnicas de carácter estadístico, de empleo de algoritmos matemáticos, de generación de reglas y de redes neuronales para el tratamiento de registros. Para otro tipo de elementos a agrupar o segmentar, como texto y documentos, se usan técnicas de reconocimiento de conceptos. Esta técnica suele servir de punto de partida para después hacer un análisis de clasificación sobre los grupos.

La principal característica de esta técnica es la utilización de una medida de similaridad que, en general, está basada en los atributos que describen a los objetos, y se define usualmente por proximidad en un espacio multidimensional. Para datos numéricos, suele ser preciso prepararlos antes de realizar Minería de Datos sobre ellos, de manera que en

primer lugar se someten a un proceso de estandarización (generalmente se normalizan para eliminar las unidades de los datos) (13).

Existen varias técnicas de agrupamiento, a continuación, se abordarán tres de las más usadas (7), K-Means como algoritmo de particionado con atributos numéricos, COBWEB como método de particionado conceptual y Expectation Maximization (EM) como método de particionado probabilístico. Aunque cada uno de ellos va resolviendo problemas de los anteriores, cada uno es mejor para determinados tipos de problemas, por lo que se toman en cuenta para ser implementados en la presente investigación.

1.4.1 Algoritmo K-Means

Para la utilización de K-Means es imprescindible definir por adelantado el número de grupos que se desea (parámetro *k*), para esto se seleccionan aleatoriamente *k* elementos que representan la media o centro de cada grupo. Posteriormente cada ejemplo o instancia es asignado al grupo más cercano acorde con la distancia Euclidiana (por defecto), de Manhattan o cualquier otra, que la separa de él. Para cada uno de los grupos así construidos se calcula el centroide de todas sus instancias. Estos centroides son tomados como los nuevos centros de sus respectivos grupos. Finalmente se repite el proceso completo con los nuevos centros de los grupos. La iteración continúa hasta que se repite la asignación de los mismos ejemplos a los mismos grupos, ya que los puntos centrales de los grupos se han estabilizado y permanecerán invariables después de cada iteración (4).

1.4.2 Algoritmo COBWEB

El algoritmo K-Means se enfrenta con un problema cuando los atributos no son numéricos, ya que en ese caso la distancia entre ejemplos no está tan clara. Para resolver este problema Michalski presenta en (14) la noción de agrupamiento conceptual, que utiliza para justificar la necesidad de un agrupamiento cualitativo frente al agrupamiento cuantitativo, basado en la vecindad entre los elementos de la población. En este tipo de agrupamiento una partición de los datos es buena si cada clase tiene una buena interpretación conceptual (modelo cognitivo de jerarquías). Una de las principales motivaciones de la categorización de un conjunto de ejemplos, que básicamente supone la formación de conceptos, es la predicción de características de las categorías que heredarán sus subcategorías. Esta conjetura es la base de COBWEB (15). A semejanza de los humanos, COBWEB forma los conceptos por agrupación de ejemplos con atributos similares. Representa los grupos como una distribución de probabilidad sobre el espacio de los valores de los atributos, generando

un árbol de clasificación jerárquica en el que los nodos intermedios definen subconceptos (16). El objetivo de COBWEB es hallar un conjunto de clases o grupos (subconjuntos de ejemplos) que maximice la utilidad de la categoría (partición del conjunto de ejemplos cuyos miembros son clases) (4).

1.4.3 Algoritmo EM

Los dos algoritmos de agrupamiento anteriores presentan ciertos inconvenientes, entre los que destacan la dependencia que tiene el resultado del orden de los ejemplos y la tendencia de estos algoritmos al sobreajuste. Una aproximación estadística al problema del agrupamiento resuelve estos problemas. Desde este punto de vista, lo que se busca es el grupo de grupos más probables, dados los datos, es decir, tomar en cuenta que los ejemplos tienen ciertas probabilidades de pertenecer a un grupo. La base de este tipo de agrupamiento se encuentra en el modelo estadístico llamado *mezcla de distribuciones*. Cada distribución representa la probabilidad de que un objeto tenga un conjunto particular de pares *atributo-valor*, si se supiera que es miembro de ese grupo. Se tienen *k* distribuciones de probabilidad que representan los *k* grupos. El algoritmo EM empieza prediciendo los parámetros de las distribuciones (dicho de otro modo, se comienza pronosticando las probabilidades de que un objeto pertenezca a una clase) y, a continuación, los utiliza para calcular las probabilidades de que cada objeto pertenezca a un grupo y usa esas probabilidades para re-estimar los parámetros de las probabilidades, hasta converger (17).

1.5 Agentes y sistemas multiagentes

Para dar solución a problemas de Minería de Datos, se viene utilizando desde hace algunos años el paradigma de agentes. Se denomina agente a cualquier ente capaz de percibir su medioambiente con la ayuda de sensores y actuar en ese medio utilizando actuadores (18). Un agente debe ser capaz de razonar, pero de los agentes informáticos se espera que tengan otros atributos que los distingan de los “programas” convencionales, como que estén dotados de controles autónomos, que perciban su entorno, que persistan durante un período de tiempo prolongado, que se adapten a los cambios y que sean capaces de alcanzar objetivos diferentes. Un agente racional es aquel que actúa con la intención de alcanzar el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado (19). También se puede definir a un agente inteligente como un sistema computacional capaz de

actuar de manera independiente como representante de su usuario (satisface objetivos de diseño y sin supervisión) (20).

Un sistema multiagente es la fusión de varios agentes inteligentes que interactúan entre sí, teniendo como responsabilidad una tarea específica, en virtud de los objetivos establecidos (21).

Una de las características de un SMA es que el grupo de agentes que lo integran deben trabajar de manera conjunta e individual. De manera conjunta, para cumplir tareas globales que surgen a consecuencia de la búsqueda de solución de un problema general; y de manera individual, porque las tareas globales son descompuestas en sub-tareas, generando tareas específicas para cada uno de los agentes que participarán en la solución del problema (2).

1.6 Herramientas y Metodología

Con la finalidad de implementar una solución informática que satisfaga el problema planteado, se realiza un análisis de la metodología y herramientas necesarias.

1.6.1 Herramientas utilizadas en la construcción del sistema multiagente

Para el desarrollo de sistemas multiagentes existen diversas herramientas, entre ellas se encuentran Java Framework for Multi-agent Systems (JAFMAS), Multi-agent Development Kit (MADKit), ZEUS y JADE. Se ha escogido JADE porque su instalación es factible, posee una documentación on-line completa y soporta cualquier tipo de agente (22), entre otras características que se expondrán a continuación.

JADE es un conjunto de herramientas para la creación de aplicaciones y sistemas multiagentes bajo los estándares Foundation for Intelligent Physical Agents (FIPA) para el desarrollo de agentes inteligentes (23). Es un software libre con código fuente bajo Lesser General Public License (LGPL) desarrollado totalmente en lenguaje Java lo que permite su portabilidad y movilidad a cualquier Sistema Operativo (SO). Además, posibilita que la Plataforma de Agentes pueda estar distribuida a través de una red de computadoras, que no necesariamente deben tener el mismo SO y puede ser controlada remotamente. Posee una arquitectura flexible y eficiente en la comunicación de los agentes a través de mensajes ACL, donde JADE se encarga de crearlos y manejarlos a través de una cola de mensajes entrantes la cual es privada para cada agente (24).

El ciclo de vida de un agente JADE sigue el ciclo propuesto por FIPA. Estos agentes pasarán por diferentes estados definidos como:

- Initiated (Iniciado): El agente se ha creado, pero no se ha registrado todavía con el AMS (Agent Management System, agente que se encarga de supervisar el control de acceso y uso de la Plataforma de Agentes).
- Active (Activo): El agente ya ha sido registrado y posee nombre. En este estado se puede comunicar con otros agentes.
- Suspended (Suspendido): El agente se encuentra parado porque su hilo de ejecución se encuentra suspendido.
- Waiting (Esperando): Se encuentra bloqueado a la espera de un suceso.
- Deleted (Eliminado): El agente ha terminado, por tanto, el hilo terminó su ejecución y ya no estará más en el AMS.
- Transit (Tránsito): El agente está migrando a una nueva ubicación.

Debido a las características de portabilidad y movilidad, los paquetes de JADE pueden ser utilizados desde su interfaz principal (Remote Monitoring Agent, RMA) o desde otras aplicaciones.

La selección de Java como lenguaje de programación se basa fundamentalmente en su utilización por JADE para el desarrollo del sistema multiagente. Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, es decir, que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java hasta 2016, es uno de los lenguajes de programación más usados atendiendo a (25).

El Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) a emplear es NetBeans IDE en su versión 8.0.1 del año 2015, se selecciona debido a que este es hecho principalmente para el lenguaje de programación Java, a pesar de que puede servir para otros lenguajes de programación. Existe un número importante de módulos para extenderlo. Es un producto libre y gratuito sin restricciones de uso. Además posee gran éxito y cuenta con una cuantiosa base de usuarios, una comunidad en constante crecimiento (26). La selección del IDE se basa además en las posibilidades que ofrece para lograr la integración de JADE y Weka.

Weka es una herramienta basada en técnicas de aprendizaje automático, útil para la aplicación de la MD. Desarrollada en Java, es un software libre distribuido bajo la licencia

GNU General Public License (GNU-GPL). Constituida por paquetes que facilitan técnicas de pre-procesado, clasificación, agrupamiento, asociación y visualización. Dichos paquetes pueden ser usados desde la interfaz que provee la herramienta o bien como parte de cualquier otra aplicación. Funciona en los sistemas operativos Windows, GNU/Linux y Mac (27). Weka permite el trabajo con varios tipos de archivos como "*.m*", "*.csv*" y distintas bases de datos.

La gestión de bases de datos se realiza con PostgreSQL, este es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (28). Además, se encuentra entre los gestores más utilizados atendiendo a (29). Las características antes expuestas y la facilidad de integración con el IDE seleccionado han permitido seleccionar a PostgreSQL para el manejo de ficheros SQL.

1.6.2 Metodología de desarrollo

Las técnicas convencionales de ingeniería no tienen en cuenta las necesidades de especificación de los sistemas multiagentes, como la especificación de planificación de tareas, el intercambio de información con lenguajes de comunicación orientados a agentes, la movilidad del código o motivación de los componentes del sistema. Por ello, se plantean nuevas metodologías basadas en agentes como Beliefs, Desires and Intentions (BDI), Vowel Engineering, MAS-CommonKADS, GAIA, MESSAGE, ZEUS, Agent Unified Modeling Language (AUML), TROPOS, entre otras. INGENIAS es una metodología que se deriva de MESSAGE, considerada una evolución de las ideas de la misma, defiende las entidades agente y organización como las principales del sistema multiagente (30).

Dentro de los puntos de vista de MESSAGE se pueden encontrar los meta-modelados de Organización, Tareas y Objetivos, Agente, Dominio e Interacción, donde cada uno representa en su unión al SMA. INGENIAS extiende de MESSAGE al aportar una mayor cohesión de los modelos (de acuerdo al principio de racionalidad), al definir un nuevo modelo para definir el entorno y al considerar la integración de resultados de investigación en el área de agentes. Esta se ha construido con meta-modelos que definen cinco aspectos del sistema multiagente: Agentes, Interacciones, Organización, Entorno, Objetivos y Tareas (ver Figura 1) (30).

Para el desarrollo de cada uno de estos meta-modelos INGENIAS cuenta con su propia herramienta llamada INGENIAS Development Kit (IDK), que con los módulos apropiados permite la generación de código sobre distintas plataformas, así como generar la documentación pertinente en formato HTML.

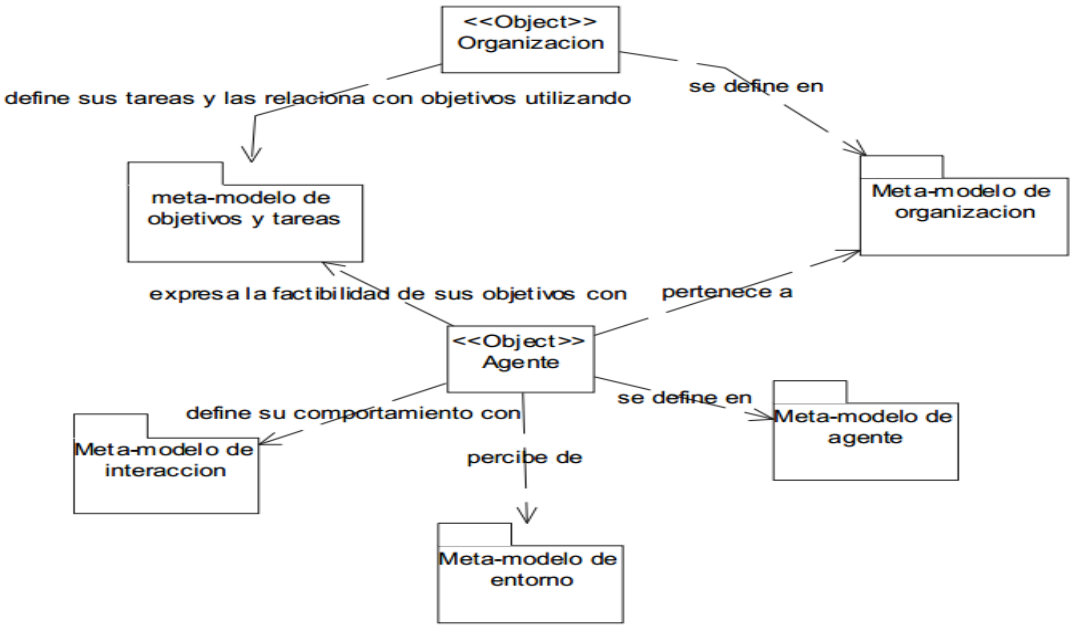

Figura 1. Relaciones entre los diferentes meta-modelos y las dos entidades principales, la organización y el agente. [fuente (31)]

El **meta-modelo de Agente** describe agentes particulares y los estados mentales en que se encuentran a lo largo de su vida, excluye las interacciones con otros agentes. Se centra en la funcionalidad del agente y en el diseño de su control. Proporciona información acerca de responsabilidades y comportamiento (32).

El **meta-modelo de Interacción** se construye sobre agentes, roles, objetivos, interacciones y unidades de interacción. En las interacciones se ejecutan unidades de interacción en las que hay un iniciador (emisor) y al menos un colaborador (receptor). Además, se justifica la participación de los actores en la interacción y la existencia de la interacción en sí mediante objetivos (33).

El **meta-modelo de Objetivos y Tareas** tiene como propósito recoger las motivaciones del SMA, definir las acciones identificadas en los modelos de Organización, Interacción o de Agente y cómo afectan estas acciones a sus responsables (34).

El **meta-modelo de Organización** es el equivalente a la arquitectura del sistema en un SMA, su valor principal son los flujos de trabajo que define. La estructura de la organización

define los elementos principales que componen la organización y cómo se construye la misma a partir de ellos (35).

En el **meta-modelo de Entorno** se distinguen tres tipos de elementos principales: agentes, recursos y aplicaciones. Se entiende por recurso a todo aquel objeto del entorno que no proporciona una funcionalidad concreta, pero que es indispensable para la ejecución de tareas y cuyo uso se restringe a consumir o restituir. Cuando el uso sea más complejo, como la funcionalidad requerida de una base de datos, se empleará el término aplicación. La denominación de agente se emplea cuando la entidad satisfaga el principio de racionalidad. Este principio expone que la entidad realiza solo aquellas acciones que le llevan a satisfacer sus objetivos (36).

Los meta-modelos son muy útiles en el proceso de desarrollo de SMA, ya que determinan qué entidades tienen que existir y cómo deben conectarse. Desde el punto de vista de ingeniería, lo que interesa de los meta-modelos, además de servir de guía, es cómo pueden ayudar a estructurar el desarrollo del SMA. INGENIAS propone la aplicación de los meta-modelos a procesos de ingeniería, tomándolos como lenguaje de especificación del SMA, de forma similar a los diagramas Unified Modeling Language (UML), como especificación de un desarrollo orientado a objetos. Como aplicación práctica de esta idea INGENIAS plantea la integración con Rational Unified Process (RUP) en las fases de análisis y diseño (ver Tabla 1).

Tabla 1. Asociaciones entre los elementos de RUP y entidades de INGENIAS.

Entidad INGENIAS	Entidad RUP
Agente	Clase
Organización	Arquitectura
Grupo	Subsistema
Interacción	Escenario
Roles, tareas y flujos de trabajo	Funcionalidad

El agente, como la clase, define tipos. Lo que se ha denominado *agente en ejecución* es un objeto en RUP. La organización equivale a la arquitectura en RUP por su carácter estructurador. La organización brinda una visión global del sistema agrupando agentes, roles, recursos y aplicaciones y estableciendo su participación en los flujos de trabajo del SMA. El grupo es la unidad de estructuración utilizada en la organización. Su similitud con un subsistema se debe a que como este, se utiliza para organizar elementos en unidades de abstracción mayores y define un conjunto de interfaces para interaccionar, los roles en

este caso. La interacción es una generalización de los diagramas de colaboración y secuencia. En RUP, los diagramas de secuencia y colaboración representan escenarios que describen cada caso de uso. Por último, roles, tareas y los flujos de trabajo proporcionan el encapsulamiento de acciones que en RUP brindan métodos e interfaces. De acuerdo con estas equivalencias y el proceso de RUP, la generación de modelos se estructura según indica la Tabla 2.

Tabla 2. Adecuación de las etapas de RUP a los meta-modelos presentados. [fuente (37)]

		FASES		
		Inicio	**Elaboración**	**Construcción**
FLUJOS DE TRABAJO FUNDAMENTALES	**Análisis**	o Generar casos de uso e identificar realizaciones de los casos de uso con modelos de interacciones. o Esbozar la arquitectura con un modelo de organización. o Generar modelos del entorno para trasladar la captura de requisitos a los modelos	o Refinar casos de uso. o Generar modelos de agente para detallar los elementos de la arquitectura. o Continuar con los modelos de organización identificando flujos de trabajo y tareas. o Modelos de tareas y objetivos para generar restricciones de control (objetivos principales, descomposición de objetivos) o Refinar modelo de entorno para incluir nuevos elementos.	o Estudiar resto de casos de uso.
	Diseño	o Generar un prototipo con herramientas de prototipado rápido, como ZEUS o Agent Tool	o Centrar el modelo de organización en el desarrollo de flujos de trabajo. o Llevar las restricciones identificadas a modelos de tareas y objetivos para dar detalles acerca de las necesidades y resultados de las tareas y su relación con los objetivos del sistema. o Expresar la ejecución de tareas dentro de modelos de interacción. o Generar modelos de agente para detallar *patrones de estado mental*.	o Generar nuevos modelos de agente o refinar los existentes. o Depurar la organización centrando el desarrollo en las relaciones sociales.

Las fases de pruebas e implementación no se han incluido. La fase de pruebas no tiene porqué ser diferente de la del software convencional. El software final ha de satisfacer casos de uso identificados, y como indica la Tabla 2 , el concepto de casos de uso se mantiene. En cuanto a la implementación, se admite que el diseñador puede actuar como en diseños usuales, desarrollando elementos computacionales que hagan lo que está especificado (37).

1.7 Consideraciones finales del Capítulo

A partir del análisis de los aspectos teóricos de la investigación se determinó trabajar con los algoritmos K-Means, COBWEB y EM para el agrupamiento y con KNN, J48, NaiveBayes y SMO para la clasificación, por ser representativos de los distintos tipos de agrupamiento y clasificación respectivamente.

Para el diseño del SMA se reconoce en la bibliografía a la biblioteca JADE como la más completa que sigue el estándar FIPA, por lo que se selecciona para implementar el sistema propuesto. Esta selección contribuye a la selección del lenguaje Java y el IDE Netbeans.

El gestor de bases de datos PostgreSQL es el más potente dentro del software libre, por lo que se selecciona para utilizar bases de datos como fuente de acceso del sistema.

La metodología INGENIAS es la más completa encontrada en la bibliografía para el desarrollo de sistemas multiagentes, por lo que será la que guiará el proceso de análisis y diseño de la aplicación.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En el presente capítulo se define el proceso de integración de las herramientas seleccionadas para el desarrollo de la propuesta de solución. Se especifican detalles de los procesos de transformación de bases de datos en PostgreSQL a archivos ARFF y de la comparación de los algoritmos de clasificación y agrupamiento. Además, se identifican los elementos de Ingeniería de Software a tener en cuenta en las fases de análisis y diseño del sistema.

2.1 Integración de herramientas con NetBeans

Para el inicio del desarrollo de la propuesta de solución fue necesario en primer lugar lograr integrar las herramientas JADE y Weka con el entorno de desarrollo. La utilización en ambas del lenguaje de programación Java, posibilitó usarlas a través de la creación de bibliotecas en el menú *Tools->Libraries* (ver Figura 2 y Figura 3) del IDE, para utilizarlas como parte de cualquier proyecto creado en NetBeans. El sistema puede visualizar información en forma de gráficos gracias a la biblioteca *JFreeChart 1.0.19* para la mejor comprensión de datos numéricos, y necesita de la biblioteca *JTattoo 1.6.11* que provee los estilos visuales de las diferentes interfaces del sistema.

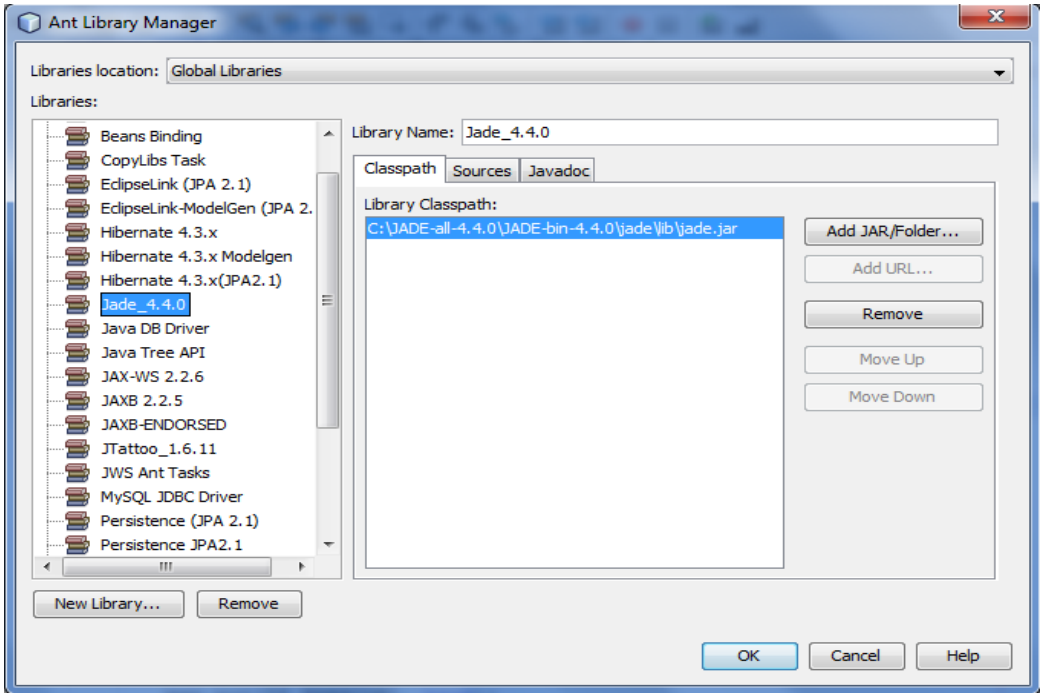

Figura 2. Creación de biblioteca de JADE.

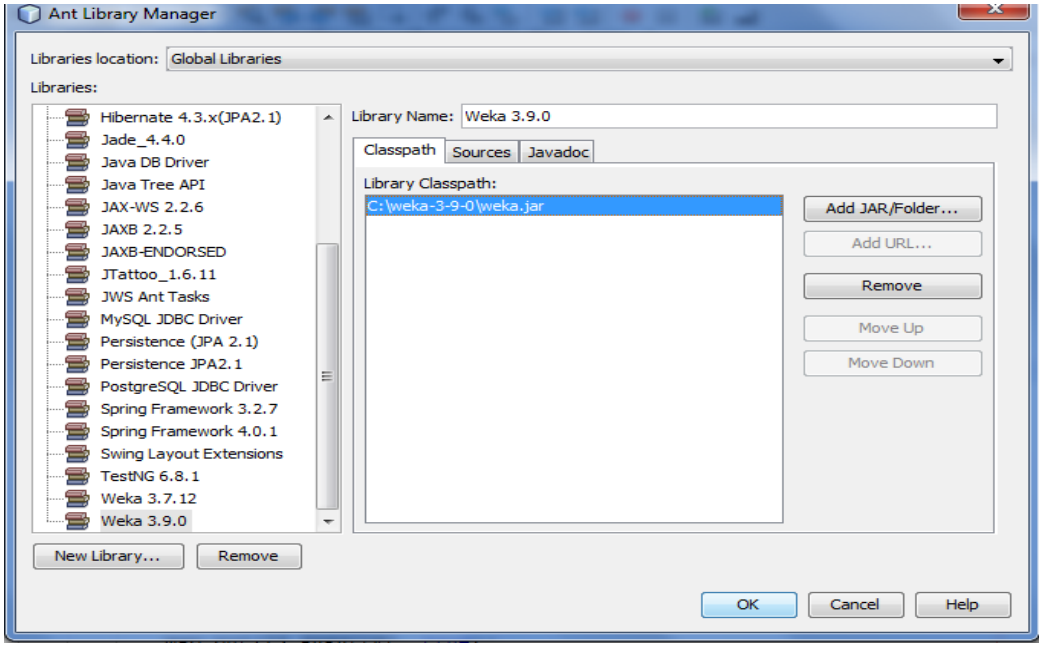

Figura 3. Creación de biblioteca de Weka.

Weka proporciona la opción de conexión con bases de datos, una de las principales problemáticas surge precisamente con dicha conexión, pues en versiones posteriores a la 7.4.x de PostgreSQL presenta no conformidades, en ocasiones con el driver y en otras no encuentra sus clases. Para solventar esta situación en la propuesta de solución se utiliza el driver JDBC, el mismo permite la ejecución de operaciones sobre bases de datos desde el lenguaje Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se acceda. Específicamente el driver utilizado fue *postgresql-9.2-1002.jdbc4*, el cual se encuentra disponible en NetBeans. Para efectuar una conexión es necesario especificar algunos campos como el host, puerto, usuario, contraseña y nombre de la base de datos.

2.2 Diseño de la transformación de bases de datos de PostgreSQL en archivos ARFF

En la introducción de la presente investigación se mencionó que Weka trabaja con los archivos en el formato ARFF, de ahí la necesidad de transformar las bases de datos proporcionadas por PostgreSQL a dicho formato. Para su ejecución los autores analizaron los diferentes tipos de datos de las tablas en PostgreSQL, con sus equivalentes en Java y luego los asociaron con los aceptados en ARFF (ver Tabla 3).

Tabla 3. Asociación de tipos de datos entre PostgreSQL, Java y Atributos de Weka.

Tipo de PostgreSQL	Tipo de Java	Tipo de atributo Weka
Char	String	nominal
char[]		
character varying		
character varying[]		
character[]		
Text		
Boolean	Boolean	
Bit		
double precisión	double	numeric
Numeric		
Smallint	short	
Integer	Int	
Bigint	Long	
Real	Float	
Date	Date	date
Time	Time	
Text	Text	string

El proceso de transformación (ver Figura 4) se divide en dos etapas, la conexión a la base de datos y la creación del archivo ARFF. La primera etapa inicia con la recolección de los datos necesarios para la conexión con PostgreSQL, los cuales fueron mencionados con antelación, luego finaliza con la selección de la tabla con la que se desea trabajar o la ejecución de una consulta SQL. La segunda etapa consiste en preparar los datos para la confección del archivo, para ello se recoge el nombre de la relación, los atributos a usar y los tipos correspondientes, y, por último, los datos. Al finalizar la segunda etapa se cuenta con la información necesaria para conformar el archivo ARFF y almacenarlo en una ubicación escogida por el usuario.

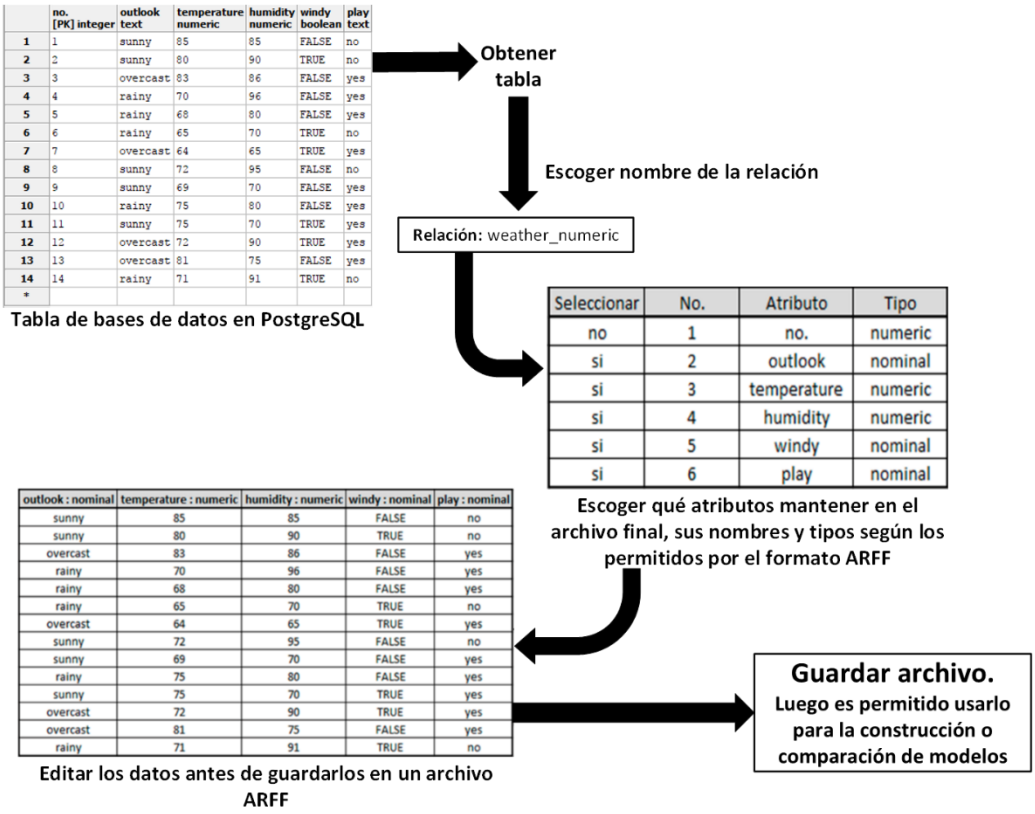

Figura 4. Proceso de transformación de una tabla en PostgreSQL a un archivo ARFF.

2.3 Comparación de algoritmos

La realización de la comparación de los algoritmos de clasificación y agrupamiento seleccionados es la función principal de la propuesta de solución. Esta operación le permite al sistema sugerirle al usuario el mejor algoritmo clasificador o agrupador ante un problema determinado, atendiendo a un conjunto de métricas.

2.3.1 Clasificación

La evolución de la evaluación de los clasificadores depende de la denominada matriz de confusión (ver Tabla 4), esta permite visualizar a través de una tabla de contingencia la distribución de los errores cometidos por los algoritmos, donde los Verdaderos Positivos son instancias correctamente reconocidas por el sistema, los Falsos Negativos son instancias positivas y que el sistema no reconoce como tales, los Falsos Positivos son instancias negativas pero el sistema dice que no lo son y los Verdaderos Negativos son instancias negativas y correctamente reconocidas (38).

Tabla 4. Ejemplo de Matriz de Confusión.

	Clase predicha	
Clase real	X	Y
X	Verdaderos Positivos (VP)	Falsos Negativos (FN)
Y	Falsos Positivos (FP)	Verdaderos Negativos (VN)

La matriz de confusión posibilita el cálculo de algunas métricas como por ejemplo la cantidad de instancias clasificadas correcta e incorrectamente, porcentaje de exactitud, costo total y promedio, precisión, recall, estadístico *kappa* de Cohen, Medida-F, especificidad, AUC y coeficiente de correlación de Matthews o Phi.

La cantidad de instancias clasificadas correcta e incorrectamente son determinadas por las ecuaciones 1 y 2 respectivamente. El porcentaje de exactitud es el porciento que representa la cantidad de instancias clasificadas correctamente con respecto al total de instancias (*N*), como se refleja en la ecuación 3.

$$Correctos = VP + VN \qquad (1)$$

$$Incorrectos = FP + FN \qquad (2)$$

$$\%Exactitud = \frac{Correctos \times 100}{N} \qquad (3)$$

La precisión es la proporción de casos positivos predichos que son realmente positivos (39) y puede calcularse a través de la ecuación 4, mientras que recall es la razón de casos reales positivos predichos como positivos de manera correcta y se determina mediante la ecuación 5. La precisión y el recall mantienen una relación inversa, cuando intenta aumentar el recall, la precisión tiende a disminuir y viceversa. La Medida-F es una media equilibrada entre la precisión y el recall, esta se puede computar siguiendo la ecuación 6.

$$Precision = \frac{VP}{FP + VP} \qquad (4)$$

$$Recall = \frac{VP}{FN + VP} \qquad (5)$$

$$Medida\_F = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (6)$$

El estadístico *kappa* de Cohen es una métrica que compara una precisión observada con una precisión esperada. El cálculo de la precisión observada y la precisión esperada es

parte integral de la comprensión del estadístico *kappa*, y se ilustra con mejor claridad mediante el uso de una matriz de confusión (40). Para lograr una mayor comprensión del cálculo del estadístico se utilizará la matriz de confusión mostrada en la Tabla 5. En la matriz de confusión se observa que hay 30 casos totales (10 + 7 + 5 + 8 = 30). De acuerdo con la primera columna 15 se etiquetaron como A (10 + 5 = 15), y de acuerdo con la segunda columna 15 se etiquetaron como B (7 + 8 = 15). También se aprecia que el modelo clasificó 17 casos como A (10 + 7 = 17) y 13 casos como B (5 + 8 = 13).

Tabla 5. Ejemplo de Matriz de Confusión para el cálculo del estadístico kappa.

	Clase real	
Clase predicha	A	B
A	10	7
B	5	8

La precisión observada (P_o) es simplemente el número de casos que se clasificaron correctamente a lo largo de toda la matriz de confusión, es decir, el número de casos que fueron etiquetados como A o B, tanto en los casos reales como en los predichos por el clasificador. Para calcular la precisión observada, se agrega el número de instancias en las que el clasificador estuvo de acuerdo con la etiqueta real y se divide por el número total de instancias. Para esta matriz de confusión, la precisión observada sería $\mathrm{P_o} = \frac{10+8}{30} = 0.6$.

La precisión esperada (P_e) está directamente relacionada con el número de casos de cada clase (A y B), junto con el número de casos en los que el clasificador estuvo de acuerdo con la etiqueta real. El cálculo de la precisión esperada para el ejemplo tratado, se realiza, primero, multiplicando la frecuencia marginal de la clase real A por la frecuencia marginal de la clase predicha A, y dividiéndolo por el número total de instancias. La frecuencia marginal de una cierta clase es justamente la suma de todas las instancias, reales o predichas, para esa clase. En este caso, 15 (10 + 5 = 15) instancias fueron etiquetadas como A de acuerdo a la clase real y 17 (10 + 7 = 17) de acuerdo a la clase predicha, obteniendo como resultado un valor de $\frac{15 \times 17}{30} = 8.5$. Luego, se aplica el mismo procedimiento para la clase B (y se puede repetir para cada clase adicional si hay más de 2), 15 (7 + 8 = 15) casos fueron etiquetados como B de acuerdo a la clase real y 13 (5 + 8 = 13) acorde a la clase predicha. Alcanzando como resultado un valor de $\frac{15 \times 13}{30} = 6.5$. El paso final es sumar todos los valores obtenidos, y dividirlos por el número total de instancias, resultando en $P_e = \frac{8.5+6.5}{30} = 0.5$.

Finalmente se calcula el valor del estadístico *kappa* utilizando la ecuación 7 sustituyendo los valores de la precisión observada y esperada determinados previamente.

$$kappa = \frac{P_o + P_e}{1 - P_e} \qquad (7)$$

La especificidad mide la proporción de negativos que se identifican correctamente como tales y se determina mediante la ecuación 8.

$$Especificidad = \frac{VN}{VN + FP} \qquad (8)$$

El costo total de clasificación permite incluir información referente a las penalizaciones relativas asociadas a una clasificación incorrecta, es decir, es la suma de los Falsos Negativos y los Falsos Positivos. El costo promedio equivale al costo total dividido entre el número total de instancias.

El coeficiente de correlación de Mattews (MCC, por sus siglas en inglés) mide la calidad de clasificaciones binarias, como tiene en cuenta los verdaderos y falsos positivos y negativos, se considera generalmente una medida equilibrada que puede utilizarse incluso con más de dos clases. En esencia el MCC es un coeficiente de correlación entre las clasificaciones observadas y predichas. Devuelve un valor entre -1 y 1, donde -1 indica una mala clasificación y 1 una clasificación perfecta, es posible calcularlo mediante la ecuación 9.

$$MCC = \frac{VP \times VN - FP \times FN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}} \qquad (9)$$

Un gráfico de Características de Operación del Receptor (ROC, por sus siglas en inglés) es una técnica para visualizar, organizar y seleccionar clasificadores basados en su desempeño. Los gráficos ROC se han utilizado durante mucho tiempo en la teoría de la detección de señales para describir el equilibrio entre las tasas de verdaderos positivos (equivalente al recall) y las tasas de falsos positivos de los clasificadores. Una curva ROC es una representación bidimensional del rendimiento del clasificador. Para comparar clasificadores, se recomienda reducir el rendimiento de ROC a un solo valor escalar que representa el rendimiento esperado. Un método común es calcular el área bajo la curva ROC, abreviado AUC por sus siglas en inglés (41). Por último, se toman en cuenta el Error Absoluto Medio (EAM), la Raíz del Error Cuadrático Medio (RECM), el Error Absoluto Relativo (EAR) y la Raíz del Error Cuadrático Relativo (RECR), que son calculados durante la evaluación de los modelos por Weka.

El sistema desarrollado construye los modelos, los evalúa y posteriormente conforma una tabla con las métricas anteriormente presentadas. En dicha tabla se comprueba qué modelo

obtiene mejores resultados en cada una de las métricas. Finalmente, sugiere el clasificador que mejores resultados alcanzó en la mayor cantidad de métricas.

2.3.2 Agrupamiento

El agrupamiento es un tipo especial de clasificación, aplicado cuando no se tienen clases que predecir, por tanto, se pudieran utilizar las métricas expuestas en la comparación de los clasificadores, pero existe la interrogante de cómo saber que una instancia ha sido ubicada en el grupo correcto o si la cantidad de grupos resultantes es la adecuada. Esto interfiere a la hora de determinar los Verdaderos y Falsos Positivos y los Verdaderos y Falsos Negativos que son la base fundamental para calcular las métricas mencionadas previamente. Una evaluación de agrupamiento demanda una medida independiente y confiable para la valoración y comparación de los experimentos de agrupamiento y sus resultados. Al comparar agrupadores, en un primer momento se pueden considerar los criterios usados por estos para formar los grupos, pero cada algoritmo adopta un criterio diferente en el momento de formar dichos grupos, haciendo imposible el uso de estos criterios. De igual forma no se puede utilizar la cantidad de grupos formados debido a que en el caso del algoritmo K-Means es necesario determinarlo antes de su ejecución y en el caso de EM puede ser determinado o no. Por ser el agrupamiento una técnica no supervisada es subjetivo a los propósitos del usuario, por lo que establecer qué agrupador brinda la mejor solución a un problema dado está muy relacionado con el criterio del usuario. No existe un esquema absoluto para medir los agrupamientos, pero para agrupar adecuadamente las instancias se debe determinar lo similares o disimilares (divergentes) que son entre sí. Para calcular lo similares o divergentes que son las instancias existe una gran cantidad de índices de similitud y disimilitud, la mayor parte son indicadores basados en la distancia, donde un elevado valor de la distancia entre dos instancias indica un alto grado de divergencia entre ellas.

Para medir la calidad del agrupador se emplean dos tipos de métricas, las externas y las internas. Las externas son medidas supervisadas que utilizan criterios no inherentes al conjunto de datos, estas comparan los resultados del agrupador contra información suministrada a priori por el usuario, utilizando ciertas medidas de calidad de agrupamiento. Dentro de las medidas externas se encuentran las expuestas en el epígrafe 2.3.1, incluyendo además la pureza entre cada grupo y la pureza general de todos los grupos, la información mutua, el índice de Jaccard, el índice Rand y el índice de Fowlkes-Mallows. Las medidas internas determinan la bondad de un agrupador considerando cuán bien

separa y compacta los grupos. Entre estas se encuentran la cohesión, la separación y el coeficiente Silhouette (Silueta, en español) utilizando para el cálculo de las mismas como medida de similitud la distancia, en el caso del sistema propuesto, la distancia euclidiana.

Cuando se trabaja con agrupadores, los problemas fundamentales que se pueden plantear son: en qué medida representa la estructura final obtenida, las similitudes o diferencias entre los objetos de estudio y cuál es el número idóneo de grupos que mejor representa la estructura natural de los datos.

En respuesta al primer problema es posible emplear las métricas internas mencionadas o como técnica tomar varias sub-muestras de la muestra original y repetir el análisis sobre cada una. Si tras repetir el análisis sobre ellas se consiguen soluciones aproximadamente iguales y parecidas a la obtenida con la muestra principal, se puede intuir que la solución obtenida puede ser válida, aunque no sería argumento suficiente para adoptar tal decisión. No obstante, este método es más útil emplearlo de forma inversa, en el sentido de que si las soluciones obtenidas en las diversas sub-muestras no guardan una cierta similitud, entonces parece evidente que se debiera dudar de la estructura obtenida con la totalidad de la muestra.

En cuanto al segundo problema existen diversos métodos para determinar el óptimo número de grupos que mejor representa a los datos naturalmente, tal es el caso de los métodos empíricos, de codo y de validación cruzada. La propuesta de solución utiliza el método empírico para determinar el máximo número de grupos en que se pueden dividir los datos, este define el cálculo de dicho número como se muestra en la ecuación 10, siendo *k* el número de grupos y *n* la cantidad de datos. Antes de realizar la comparación de los agrupadores, el sistema, utilizando el método empírico, propone al usuario usar ese número de grupos en la configuración de K-Means o mantener la opción predeterminada de 2 o un valor deseado por el usuario. Para el caso de la evaluación basada en clases se propone usar el número de clases como número de grupos para el algoritmo K-Means. Esta evaluación basada en clases posibilitará al usuario calcular todas las medidas externas e internas descritas con anterioridad, si es escogida cualquier otra forma de evaluación solo se tendrán en cuenta la medida interna del coeficiente Silhouette y la cantidad de instancias no agrupadas.

$$k \approx \sqrt{\frac{n}{2}} \qquad (10)$$

Para el cálculo del coeficiente Silhouette hay que contar con la separación y la cohesión de cada elemento. La cohesión mide qué tan cercanamente relacionado está un elemento

con los otros elementos agrupados en su mismo grupo, y la separación, qué tan distinto o bien separado está dicho elemento de otros agrupados en distintos grupos. En otras palabras, la cohesión mide la similitud dentro de un grupo, mientras la separación mide la distancia a la que se encuentran dichos grupos. La meta con estas métricas es minimizar la distancia entre los elementos de un mismo grupo y maximizar la distancia entre grupos diferentes. La cohesión y la separación de un elemento O se puede computar mediante las ecuaciones 11 y 12 respectivamente, donde $dist(O, O')$ es la distancia entre O y O', $|C_i|$ representa la cantidad de elementos agrupados dentro del grupo C_i. El coeficiente Silhouette combina en la ecuación 13 la idea tanto de la separación como de la cohesión, pero solo para un elemento O. El valor del coeficiente Silhouette puede oscilar entre -1 y 1. Un valor negativo no es deseado porque indica que la cohesión es mayor que la separación, por tanto, se busca un valor positivo (la cohesión menor que la separación), y para valores de la cohesión tan cercanos a 0 como sea posible, el coeficiente asume valor 1 cuando la cohesión es 0. Es posible calcular el ancho medio de silueta (en inglés, Average Silhouette Width) para grupos y agrupadores. En la ecuación 14 se muestra cómo calcular el ancho medio de silueta para el grupo C_i, mientras que en la ecuación 15, cómo determinarlo para el agrupador con *k* grupos (42).

$$P(C_r) = \frac{1}{n_r}\, max\, n_r^i \qquad (11) \qquad\qquad Purity(C) = \frac{1}{k}\sum_{r=1}^{k} P(C_r) \qquad (12)$$

$$a(O) = \frac{\sum_{O' \in C_i, O \neq O'} dist(O, O')}{|C_i| - 1} \qquad (13)$$

$$b(O) = min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{O' \in C_j} dist(O, O')}{|C_j|} \right\} \qquad (14)$$

$$s(O) = \frac{b(O) - a(O)}{max\{a(O), b(O)\}} \qquad (15)$$

Como anteriormente se mencionó, el cálculo de las métricas externas solo es posible para la evaluación basada en clases, donde a partir de las mismas se asigna una a cada grupo y, según esta nueva información, será posible formar una matriz de confusión semejante a la usada en la clasificación, por un lado, se encuentran las clases definidas por los datos y por otra parte los grupos. En esta matriz cada elemento X_{ji} representa la cantidad de instancias de una clase *j* asignadas a un grupo *i* (ver Tabla 6).

Tabla 6. Ejemplo de Matriz de Confusión para agrupamiento.

	Grupos					
Clases	C1	C2	C3	C4	….	Ck
L1	X11	X12	X13	X14	….	X1k
L2	X21	X22	X23	X24	….	X
L3	X31	X32	X33	X34	….	X
….	….	….	….	….	….	….
Ln	Xn1	Xn2	Xn3	Xn4	….	Xnk

Las métricas externas se dividen en dos grupos, las orientadas a clasificación y las orientadas a similitud. En el primer grupo se encuentran la entropía y la pureza. Mientras que, en el segundo grupo están el índice de Jaccard, el índice de Fowlkes-Mallows, el índice Rand y el índice gamma.

La entropía representa el grado en que cada grupo está conformado por elementos de una clase. Para cada grupo se determina en primer lugar la distribución de clases de los datos (usando la matriz de confusión). Luego, usando esta distribución de clases, la entropía de cada grupo *i* es computada mediante la ecuación 16 y la entropía total del agrupador mediante la ecuación 17. En dichas ecuaciones p_{ij} representa la probabilidad de que un miembro del grupo *i* pertenezca a la clase *j* y determina $p_{ij} = m_{ij}/m_i$ donde m_{ij} es el número de objetos de clase *j* en el grupo *i* y m_i el número de objetos en el grupo *i*. El número de clases es representado por *L*, *k* es el número de grupos y *m* el total de datos. (42)

$$e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij} \quad (16) \qquad e = \sum_{i=1}^{k} \frac{m_i}{m} e_i \quad (17)$$

La pureza también indica cómo un grupo está constituido por objetos de una clase. Utilizando la misma terminología empleada para la entropía, la pureza de un grupo se determina mediante la ecuación 18 y la pureza del agrupador a través de la ecuación 19.

$$p_i = \max_i \; p_{ij} \quad (18) \qquad pureza = \sum_{i=1}^{k} \frac{m_i}{m} p_i \quad (19)$$

Las medidas externas orientadas a similitud son determinadas con respecto a matrices de similitud. En el caso del estadístico gamma (ecuación 20, donde $\bar{x}$ y $\bar{y}$ son las medias de las matrices), es el equivalente al coeficiente de correlación entre la matriz de similitud de grupos y la matriz de similitud de clases. Ambas son matrices cuadradas y simétricas con tantas filas o columnas como cantidad de instancias haya en el conjunto de datos, en caso de una similitud se coloca el valor 1 y en caso de no haberla se coloca el valor 0. La matriz

de similitud de grupos se realiza sobre la estructura de los grupos formados correspondiendo a una similitud si dos instancias *i* y *j* están en el mismo grupo (se coloca el valor 1 en la celda *ij*), hay una divergencia si dos instancias *i* y *j* no están en el mismo grupo (se coloca el valor 0 en la celda *ij*). La matriz de similitud de clases se realiza sobre la estructura de las clases estructurada de los datos correspondiendo a una similitud si dos instancias *i* y *j* están en la misma clase (se coloca el valor 1 en la celda *ij*), hay una divergencia si dos instancias *i* y *j* no están en la misma clase (se coloca el valor 0 en la celda *ij*) (42).

$$\boldsymbol{\Gamma} = correlacion(X,Y) = \frac{\sum(x-\bar{x})\,(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}} \qquad (20)$$

Luego de formadas ambas matrices de similitud es posible extraer la información mostrada en la Tabla 7 para realizar el cálculo de métricas como el índice de Jaccard (ecuación 21), el índice Rand (ecuación 22) y el índice de Folwkes-Mallows (ecuación 23). En la Tabla 7 se muestra el número de pares de elementos que están asignados a la misma clase y grupo (a), el número de pares de elementos que están asignados a la misma clase y a distintos grupos (b), el número de pares de elementos que fueron asignados al mismo grupo y diferentes clases (c), y el número de pares de elementos que están asignadas a diferentes clases y diferentes grupos (d) (42).

Tabla 7. Tabla de contingencia para determinar qué pares de elementos están en el mismo grupo y la misma clase.

	Mismo grupo	Diferente grupo
Misma clase	a	b
Diferente clase	c	d

$$Jaccard = \frac{a}{c+b+a} \qquad (21)$$

$$Rand = \frac{a+d}{a+b+c+d} \qquad (22)$$

$$FolkesMallows = \sqrt{\frac{a}{a+c} \times \frac{a}{a+b}} \qquad (23)$$

La propuesta de solución desarrollada construye los modelos, los evalúa y luego conforma una tabla con las métricas presentadas. En esta tabla se comprueba qué modelo obtiene mejores resultados en cada una de las métricas. Finalmente, sugiere el agrupador que mejores resultados alcanzó en la mayor cantidad de métricas.

2.4 Definición del sistema multiagente

El sistema propuesto está conformado por los agentes Smek, Connector, Arff, Classifier y Clusterer. Smek es el agente principal, encargado de la administración del SMA y de la interacción con el usuario. El agente Connector se responsabiliza de la primera etapa del proceso de transformación de bases de datos de PostgreSQL en archivos ARFF. El agente Arff inicia la segunda etapa de dicho proceso con la información suministrada por Connector, y también realiza tareas de edición sobre archivos ARFF. Por su parte los agentes Classifier y Clusterer ejecutan las operaciones de clasificación y agrupamiento respectivamente.

Para asociar la aplicación de los meta-modelos a procesos de ingeniería, como se mencionaba en el epígrafe 1.6.2, se toman los meta-modelos como lenguaje de especificación del SMA, de forma similar a los diagramas UML como especificación de un desarrollo orientado a objetos (43). Esta metodología no define un ciclo de vida específico para la generación de los meta-modelos, pero sí marca prioridades. Propone, durante la fase de análisis, la creación de los meta-modelos de Agente, Organización, Objetivos y Tareas y el de Entorno, posteriormente en la fase de diseño se refinan dichos meta-modelos, además se agrega la ejecución de las tareas mediante el meta-modelo de Interacción.

INGENIAS plantea el meta-modelo de Organización como el equivalente a la arquitectura del sistema. En dicho meta-modelo se representa una relación de subordinación de los agentes Arff, Connector a Smek; Classifier y Clusterer se subordinan condicionalmente a Smek. Arff además se subordina incondicionalmente a Connector. Connector consume recursos de la aplicación externa PostgreSQL. Weka como aplicación interna le provee recursos a Arff, Classifier y Clusterer (ver

Figura 5).

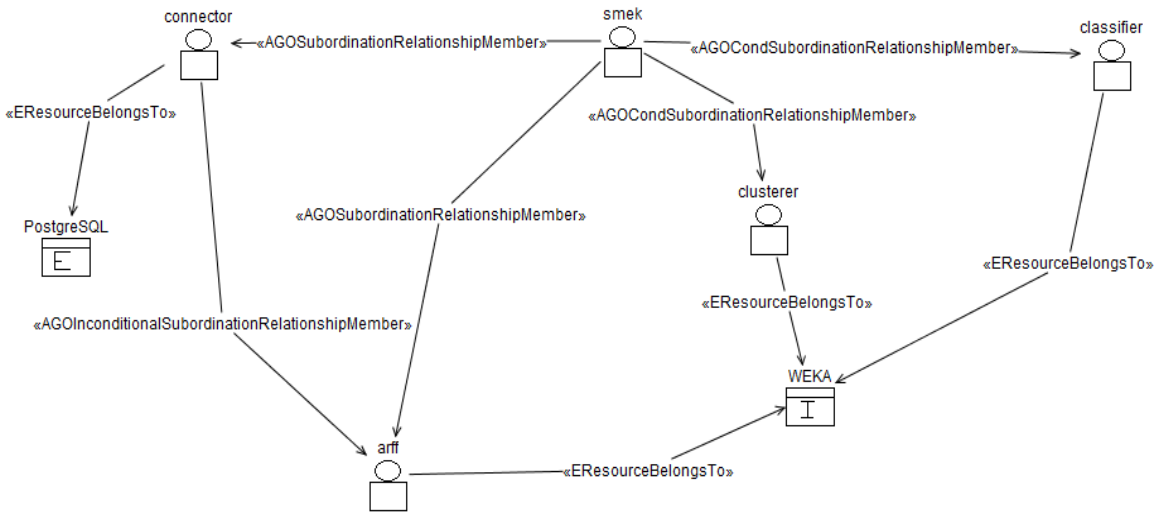

Figura 5. Meta-modelo de Organización.

2.5 Análisis y diseño del sistema

Antes de llegar al desarrollo de los procesos expuestos previamente es imprescindible abordar aspectos relacionados con la Ingeniería de Software, definida por algunos autores como el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas de software. A continuación, se definirán los requisitos funcionales y no funcionales, así como los meta-modelos que especifica la metodología seleccionada.

2.5.1 Especificación de requisitos

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina ingeniería de requerimientos.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. A menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (44). A continuación, se definen los requisitos no funcionales del sistema propuesto.

RNF1. Usabilidad:

<Requisito de Usabilidad 1.1> Especificaciones de software:

- El sistema funciona en los Sistemas Operativos Linux y Microsoft Windows 7 o superior.

- Se debe instalar la máquina virtual de Java JDK 1.7 o superior.
- Se debe instalar PostgreSQL 9.2 o superior para la gestión de bases de datos.

<Requisito de Usabilidad 1.2> Especificaciones de hardware:

- Es necesario una memoria RAM con un mínimo de 1GB.
- El microprocesador debe ser Intel Dual Core de 2.6GHz o superior.
- Es necesario una capacidad de almacenamiento mínimo de 4GB.

RNF2. Diseño e implementación:

- Para la implementación del sistema se debe usar el lenguaje de programación Java.
- El Análisis y Diseño se realiza utilizando la metodología INGENIAS.

RNF3. Rendimiento:

- El tiempo de respuesta del sistema es directamente proporcional con la cantidad de datos a procesar.

RNF4. Portabilidad:

- El sistema es compatible con los sistemas Linux y Windows, sin la necesidad de ser instalado.

Los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos también declaran explícitamente lo que el sistema no debe hacer (44). Los requisitos funcionales del sistema propuesto se encuentran especificados en la Tabla 8.

Tabla 8. Requisitos funcionales del sistema.

	Requisito Funcional	Descripción
RF1	Realizar conexión con PostgreSQL	Para realizar la conexión con PostgreSQL es necesario especificar los campos de host, puerto, usuario y contraseña.
RF2	Guardar tabla PostgreSQL como archivo ARFF	Durante el proceso es necesario elegir el nombre de la relación, los datos y tipos de datos deseados en el archivo resultante, modificar, de ser necesario algún dato vacío, y finalmente guardar el archivo ARFF.
RF3	Construir modelo de clasificación	En la construcción de los modelos de clasificación se seleccionará el archivo ARFF con el que se desea trabajar o continuar con uno guardado previamente, luego se podrá configurar el algoritmo y guardar o no el modelo construido. Después se procederá a la ejecución.
RF4	Construir modelo de clasificación en paralelo	Este proceso es similar a la construcción de los modelos de clasificación, con la variante de permitir la ejecución de varios algoritmos en paralelo.

RF5	Construir modelo de agrupamiento	En la construcción de los modelos de agrupamiento se seleccionará el archivo ARFF con el que se desea trabajar o continuar con uno guardado previamente, luego se podrá configurar el algoritmo y guardar o no el modelo construido. Después se procederá a la ejecución.
RF6	Construir modelo de agrupamiento en paralelo	Este proceso es similar a la construcción de los modelos de agrupamiento, con la variante de permitir la ejecución de varios algoritmos en paralelo.
RF7	Clasificar nuevos casos	Para la clasificación de nuevos casos se debe seleccionar el archivo ARFF y su modelo correspondiente, luego se puede realizar la clasificación.
RF8	Agrupar nuevos casos	Para el agrupamiento de nuevos casos se debe seleccionar el archivo ARFF y su modelo correspondiente, luego se puede realizar el agrupamiento.
RF9	Comparar algoritmos clasificadores	En la comparación de los algoritmos clasificadores se puede trabajar con un archivo ARFF guardado previamente o seleccionar uno distinto, configurar los algoritmos si se desea, proceder con la comparación y finalmente se podrá observar un resumen con la evaluación de los algoritmos y la sugerencia del sistema del mejor algoritmo atendiendo a las métricas escogidas.
RF10	Comparar algoritmos agrupadores	En la comparación de los algoritmos agrupadores se puede trabajar con un archivo ARFF guardado previamente o seleccionar uno distinto, configurar los algoritmos si se desea, proceder con la comparación y finalmente se podrá observar un resumen con la evaluación de los algoritmos y la sugerencia del sistema del mejor algoritmo atendiendo a las métricas escogidas.
RF11	Visualizar modelos guardados	El visor de modelos posibilita el análisis de uno o varios archivos MODEL, construidos durante los procesos de clasificación y agrupamiento.
RF12	Editar archivos ARFF	La edición de archivos ARFF brinda la opción de modificar los valores de los atributos de uno o varios archivos.

2.5.2 Meta-modelo de Agentes

INGENIAS plantea al agente como una entidad autónoma. Una Entidad Autónoma se caracteriza por tener propósitos y una identidad única. La entidad agente, por herencia de Entidad Autónoma, adquiere la capacidad de perseguir objetivos y, mediante las asociaciones con roles y tareas, de alcanzarlos. Un agente, siguiendo la definición de Newell (45) es una entidad autónoma que actúa en el nivel de conocimiento y que se basa en el principio de racionalidad mencionado previamente.

El meta-modelo contempla el uso de roles para asignar responsabilidades. Su papel consiste en separar la definición del agente de lo que se requiere de él. La relación entre roles y agentes se hace mediante la meta-relación *WFPlays*, esta indica al diseñador que el agente adquiere todos los objetivos asociados al rol, sus responsabilidades y sus

capacidades. Las relaciones entre los objetivos y el agente pueden ser GT*Pursues* (heredada de la Entidad Autónoma) y *WFPursues* (como parte de la definición de flujo de trabajo) (32). Para la solución propuesta son necesarios cinco meta-modelos de agentes. En el meta-modelo de Smek (ver Figura 6), este cumple el rol administrador y persigue cuatro objetivos, administrar agentes, interactuar con el usuario, cerrar el sistema y permitir la visualización de los modelos.

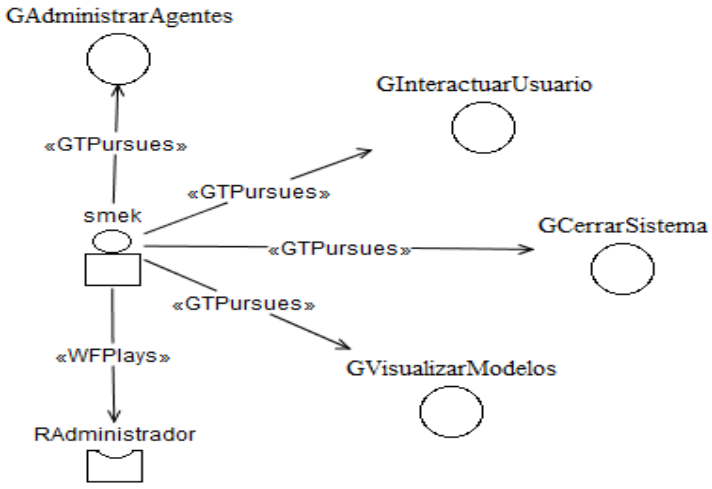

Figura 6. Meta-modelo de Agente Smek.

En el meta-modelo de Arff (ver Figura 7), dicho agente reliza el rol de Manejador de archivos ARFF, este rol hereda las características del rol Ejecutor. El agente Arff persigue un objetivo, manejar dichos archivos.

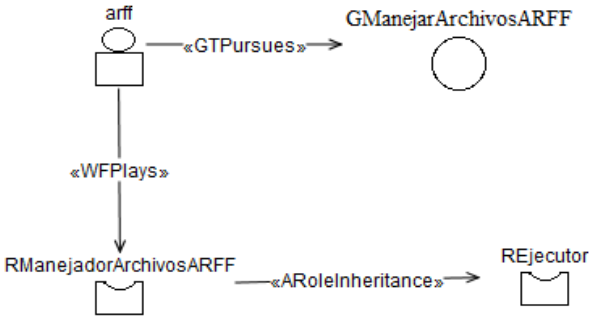

Figura 7. Meta-modelo de Agente Arff.

En el meta-modelo de Connector (ver Figura 8), este agente va a perseguir el objetivo de conectar con PostgreSQL, desempeña el rol de Conector, quien a su vez hereda las características del rol Ejecutor.

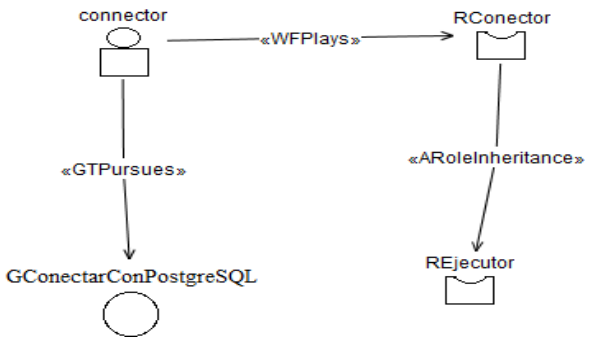

Figura 8. Meta-modelo de Agente Connector.

En el meta-modelo de Classifier (ver Figura 9), el agente realiza el rol de Clasificador y va a perseguir los objetivos de construir modelos de clasificación, clasificar nuevos casos, generar modelos de clasificadores en paralelo y comparar los algoritmos de clasificación. El rol Clasificador se relaciona con el rol Ejecutor a través de una herencia.

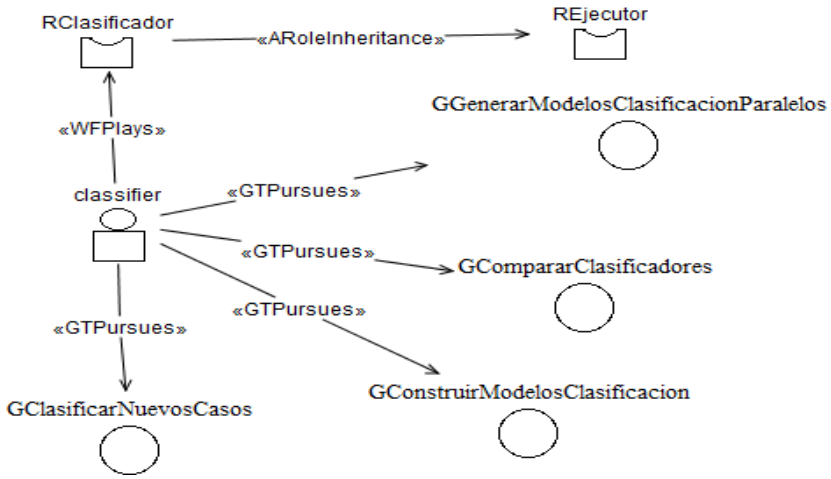

Figura 9. Meta-modelo de Agente Classifier.

En el meta-modelo de Clusterer (ver Figura 10), la entidad agente cumple el rol de Agrupador y persigue cuatro objetivos, construir modelos de agrupamiento, agrupar nuevos casos, generar modelos de agrupamiento en paralelo y comparar los algoritmos de agrupamiento. El rol Agrupador se relaciona con el rol Ejecutor a través de una herencia.

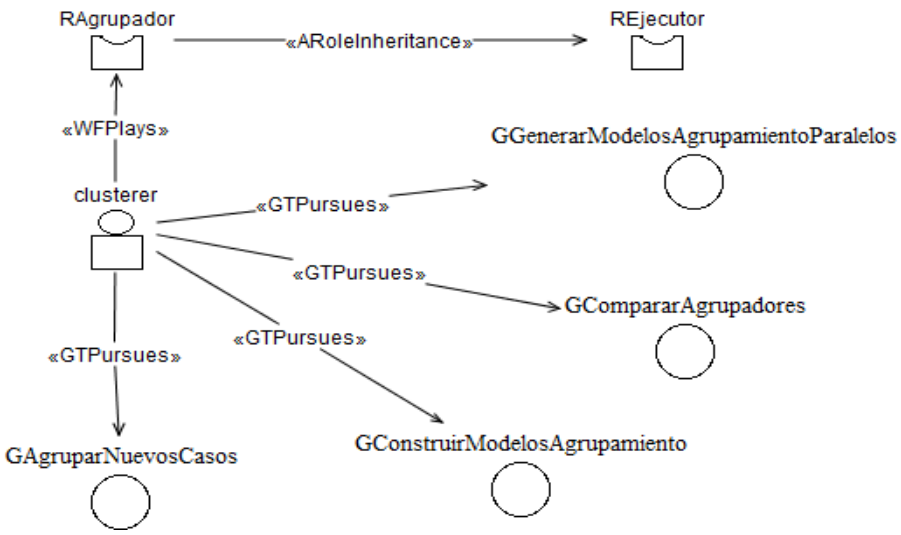

Figura 10. Meta-modelo de Agente Clusterer.

2.5.3 Meta-modelo de Objetivos y Tareas

El meta-modelo de Objetivos y Tareas trata de explicar cuáles son las consecuencias de ejecutar las tareas y porqué se deben llegar a ejecutar. Las acciones del agente se justifican por los objetivos que persigue, lo cual lleva también a la asociación de tareas y objetivos. Algunos autores definen a las tareas como procesos, mientras los objetivos se emplean para razonar acerca de las posibles alternativas que se le presentan a un agente en un momento dado. Las tareas se asocian con los objetivos mediante la meta-relación *GTAffects*, además se definen las relaciones *GTDestroys* (para destruir una o varias entidades), *GTCreates* (para crear entidades) o *GTModifies* (modificar entidades). En el caso de los objetivos se concreta la forma de modificarlos: satisfaciéndolo (*GTSatisfies*) o haciéndolo fallar (*GTFails*). Debido a la complejidad que puede existir en los objetivos y las tareas, se incorporan meta-relaciones que representan su descomposición (*GTDecomposes* y *WFDecomposes*) (34).

Luego de explicar las principales relaciones entre objetivos y tareas, se muestra uno de los ejemplos desarrollados para la propuesta de solución. En este meta-modelo (ver Figura 11) el agente Classifier persigue el objetivo de clasificar nuevos casos, para ello es responsable de la realización de la tarea con el mismo nombre, la cual se descompone en clasificar nuevos casos de SMO, J48, KNN y NaiveBayes, respectivamente. La tarea mencionada usa los recursos ArchivoMODEL y ArchivoARFF, así como métodos de Weka. Además, se encuentra asociada con la tarea TEscogerAlgoritmoClasificación, esta, a su vez depende de TRecibirConfiguración. Producto de la relación de los agentes Classifier y Smek en la

realización de la tarea TRecibirConfiguración se crea la interacción de cooperación IClasificarNuevosCasos.

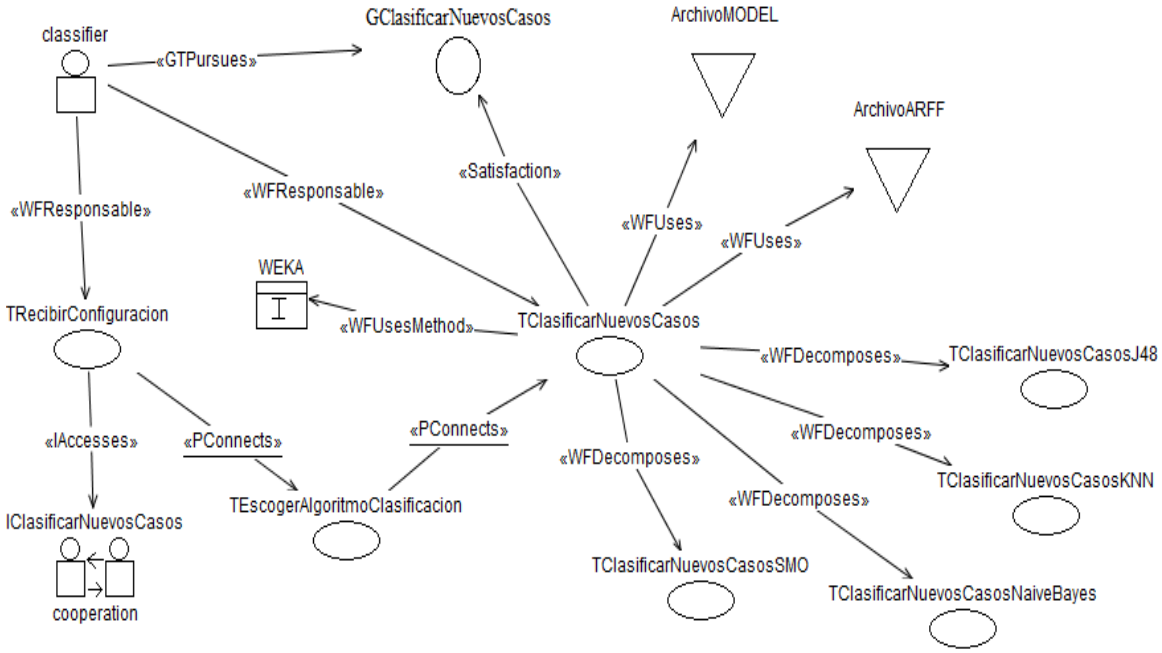

Figura 11. Meta-modelo de Objetivos y Tareas. Clasificación de nuevos casos.

2.5.4 Meta-modelo de Interacciones

Las interacciones determinan el comportamiento de los agentes mostrando cuál es su reacción cuando actúan sobre ellos. En RUP, un primer paso en el análisis es generar casos de uso que hagan referencia a interacciones clave y quizás añadir diagramas de colaboración, secuencia o de actividades para mostrar cómo evoluciona el sistema. Posteriormente, en el diseño, estas interacciones se detallan con diagramas de estado y de clase para mostrar las interfaces relevantes. En el área de agentes, aunque el análisis puede ejecutarse de una forma similar, definiendo algunos diagramas, el diseño no puede ser así. La especificación completa de una interacción tiene que cubrir, entre otros aspectos, los actores que participan, la definición de las unidades de interacción y el orden sobre dichas unidades (33).

Este meta-modelo cubre los aspectos de contexto, naturaleza, ejecución y representación. El contexto aparece como la motivación de la interacción (meta-relación *IPursues*) y de los roles en el momento de participar en la interacción. La naturaleza se asocia directamente como propiedad de la interacción. La ejecución se refiere al conjunto de actividades requeridas en el momento de desarrollar la interacción y la representación toma la

información de la ejecución y la naturaleza para crear una representación gráfica o formalismo textual sencillo de manejar por los ingenieros.

INGENIAS define dos tipos de ejecución, diagramas especializados *GRupo de Agentes Software del departamento de sistemas Informáticos y programación* (GRASIA) y diagramas de colaboración UML. En los primeros, las unidades de interacción son mensajes y la asociación entre emisor y receptor son las relaciones *IInitiates* e *IColaborates*. Las tareas pueden agruparse al igual que los agentes o roles, permitiendo el recibo de mensajes. Los diagramas de colaboración UML no están pensados para el modelado de interacciones entre agentes. La justificación de porqué se está ejecutando la interacción, se aceptan ciertos mensajes y porqué transcurre la interacción de un modo concreto, no son fácilmente expresables. Por ello, se genera una especificación GRASIA donde se tienen tareas, agentes, roles, interacciones y unidades de interacción (33).

Para la propuesta de solución se realizaron los correspondientes meta-modelos de interacción, de los cuales se presentan algunos ejemplos.

En la Figura 12 se visualiza el meta-modelo de la interacción Clasificar nuevos casos, donde dicha interacción persigue el objetivo de GClasificarNuevosCasos, para satisfacerlo se ejecuta la tarea TRecibirConfiguración. En esta interacción intervienen los roles de RAdministrador como emisor y RClasificador como receptor. Además, cuenta con una especificación UML (Figura 13) y GRASIA (Figura 14).

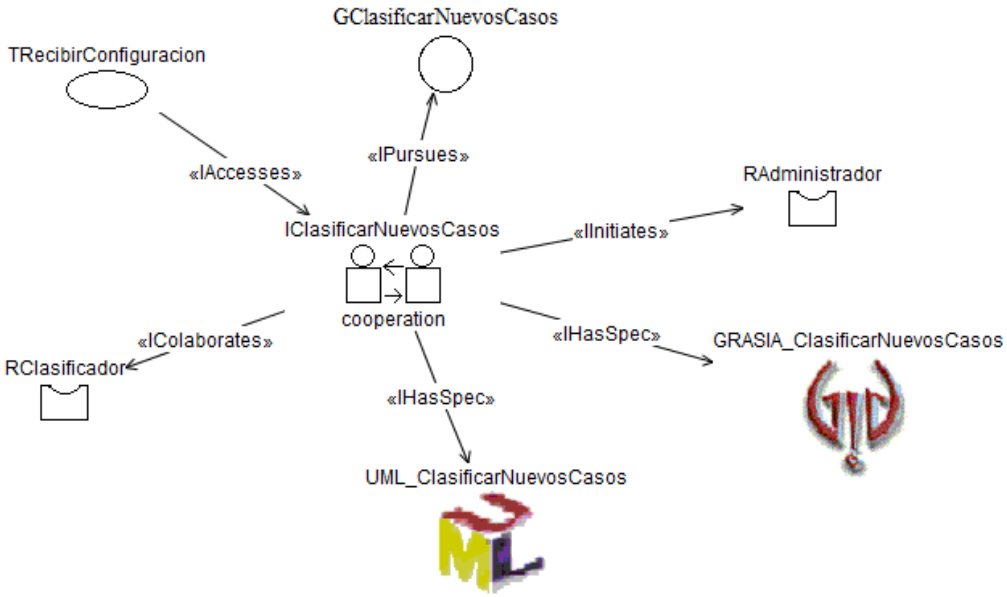

Figura 12. Meta-modelo de Interacción. Clasificar nuevos casos.

Según la especificación UML de la interacción Clasificar nuevos casos, esta comienza cuando el Administrador le envía al Clasificador las direcciones del modelo y del archivo con los nuevos casos, en ese momento el Clasificador realiza el proceso de clasificación y le envía al Administrador los resultados (ver Figura 13).

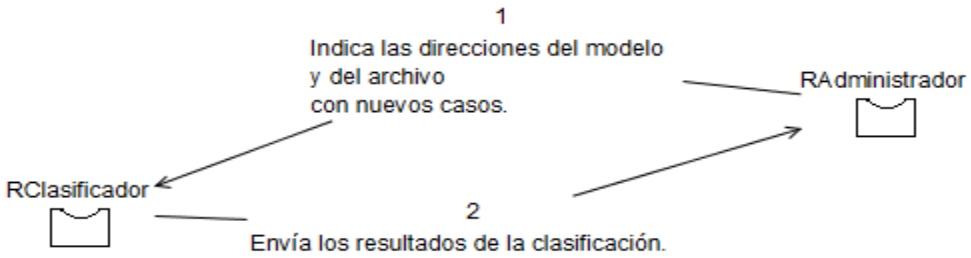

Figura 13. Especificación UML del meta-modelo de Interacción. Clasificar nuevos casos.

Según la especificación GRASIA de la interacción Clasificar nuevos casos, el acto del habla inicia con un ACL-request por parte del Administrador destinado al Clasificador, con la información a utilizar en la clasificación. Clasificador puede responder ante esta solicitud de dos formas, si entiende o no el mensaje. En caso de entender, acepta realizar la operación y responde con un ACL-agree. Si no entiende, no realiza la operación y envía al Administrador un ACL-not-understood. Luego intenta realizar la operación, si esta termina exitosamente se notifica al Administrador el resultado con un ACL-inform, sino un ACL-failure es enviado (ver Figura 14).

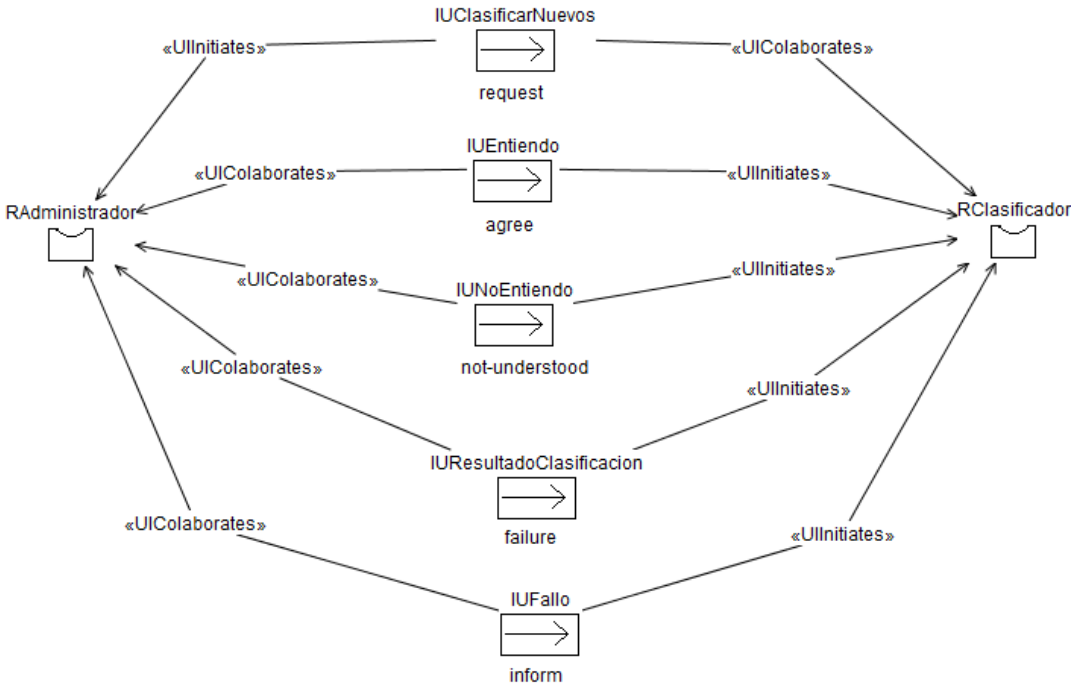

Figura 14. Especificación GRASIA del meta-modelo de Interacción. Clasificar nuevos casos.

2.5.5 Meta-modelo de Entorno

Luego de presentar la composición interna del sistema, las relaciones entre los agentes y los distintos procesos necesarios para satisfacer los objetivos trazados, en el meta-modelo de entorno se especifican las interacciones del SMA con el entorno que lo rodea. La presentación del meta-modelo de entorno comienza restringiendo el tipo de elementos que van a aparecer. Como se expresaba anteriormente se distinguen tres posibles tipos de elementos, agentes, recursos y aplicaciones. Los recursos pertenecen a un agente o a un grupo (meta-relación *EResourceBelongsTo*), las aplicaciones se caracterizan por poseer un conjunto de operaciones utilizadas para modelar la percepción del agente (meta-relación *EPerceives*) (36).

En el meta-modelo de entorno (ver Figura 15) de la propuesta de solución se aprecia cómo el agente Connector percibe de la aplicación externa PostgreSQL, mientras que de la aplicación interna Weka perciben los agentes Arff, Classifier y Clusterer. Además, se puede apreciar cómo Classifier, Clusterer y Smek necesitan usar los recursos ArchivoARFF y ArchivoModel y el agente Arff solo requiere del recurso ArchivoArff.

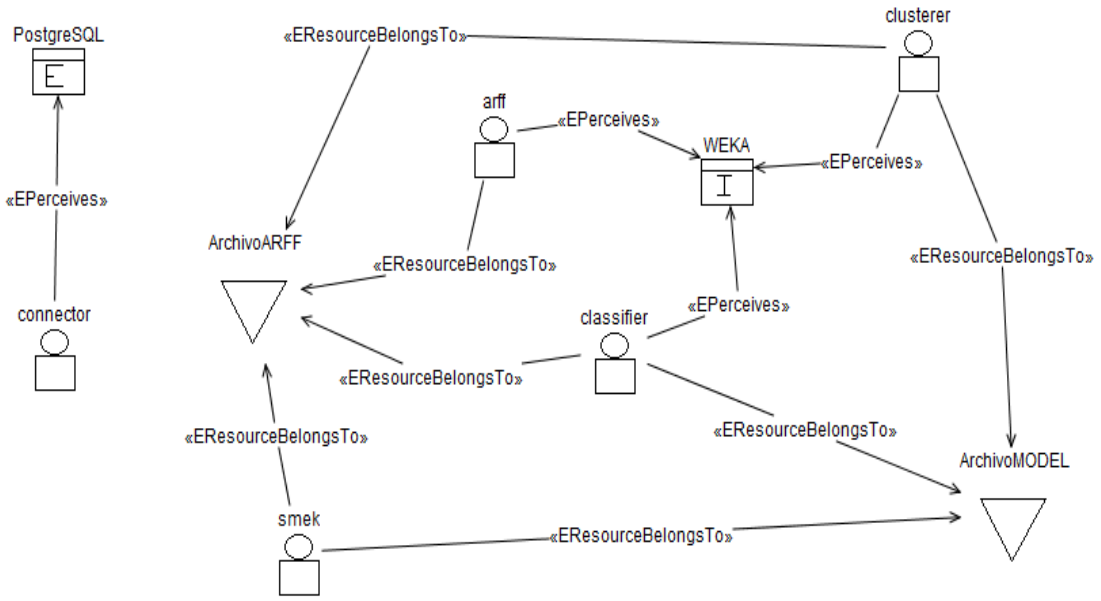

Figura 15. Meta-modelo de Entorno.

2.6 Consideraciones finales del Capítulo

Para llevar a cabo el desarrollo del sistema multiagente se lograron integrar las bibliotecas necesarias JADE y Weka con el IDE Netbeans. A partir de asociaciones entre los tipos de datos, se logró la conversión de bases de datos en PostgreSQL a archivos ARFF para poder utilizar los algoritmos implementados en Weka.

Se identificaron un conjunto de métricas que provee Weka para algoritmos de clasificación, necesarias para lograr la comparación de los algoritmos seleccionados. Fue necesario estudiar métricas para incorporar al sistema, con el objetivo de comparar los resultados de los algoritmos de agrupamiento. Por último, siguiendo la metodología INGENIAS y la descripción de sus artefactos principales, se realiza el proceso de análisis y diseño del sistema.

CAPÍTULO 3: RESULTADOS Y PRUEBAS

El presente capítulo expone los estándares de codificación empleados en el desarrollo del sistema, los resultados obtenidos durante la aplicación del proceso de trasformación de bases de datos de PostgreSQL en archivos ARFF, la comparación de los algoritmos de clasificación y agrupamiento, así como las pruebas aplicadas.

3.1 Codificación

Establecer ciertos estándares de programación facilita la comunicación de los desarrolladores y la implementación de cambios. Para la realización del sistema se definieron las siguientes pautas de codificación:

- ✓ Nombres:
    1. Los identificadores de las clases, variables, métodos, interfaces y similares deben ser escritos en el idioma inglés, solo se permite usar el idioma español en los comentarios y valores de cadenas de texto, de coincidir con palabras reservadas se permiten cambios en algunos caracteres para evitar cualquier confusión.

       Ejemplo: *class* se escribirá como *clazz*.

    2. Las clases, enums, interfaces y similares comenzarán con letra inicial mayúscula.

       Ejemplo: `public class Smek { }`

    3. Los nombres de los métodos deben comenzar con letra inicial minúscula, en caso de ser compuestos la próxima palabra debe iniciar con mayúscula.

       Ejemplo: `public void startAgent { }`

    4. Los nombres de las variables pueden comenzar con letra inicial mayúscula o minúscula indistintamente, de comenzar con mayúscula no deben contener minúsculas. También podrán comenzar con el carácter '_' y de ser así la primera letra después de '_' será tomada como la letra inicial del nombre y debe cumplir con todo lo anterior.

       Ejemplo: `private String RELATION;`

```
private int classIndex;
```

5. Todos los métodos sobrecargados deberán tener la etiqueta '*@Override*'.

Ejemplo: @Override

```
protected void setup() {   }
```

3.2 Transformación de bases de datos de PostgreSQL a archivos ARFF.

Dentro de la transformación de bases de datos en PostgreSQL a archivos ARFF intervienen los agentes Smek, Connector y Arff. Smek es el responsable de iniciar a Connector en la tarea de conexión a PostgreSQL. Connector a través de la información recogida del usuario realiza una conexión a una base de datos, luego permite escoger una de las tablas presente en la misma o realizar una consulta SQL con el objetivo de obtener una tabla, por último, indica a Arff que se inicia una transformación suministrando un objeto que contiene el nombre de la tabla, los nombres de las columnas, los tipos de datos en PostgreSQL de cada columna y los datos. Arff recibe este objeto de Connector, asocia el nombre de la tabla con el nombre de la relación de datos, los nombres de las columnas como los nombres de los atributos, asocia los tipos de datos en PostgreSQL de cada columna con sus correspondientes tipos de datos en Weka para cada atributo, revisa los datos para formar una tabla y finalmente modifica los atributos o datos editados por el usuario para formar el fichero ARFF final.

El proceso antes descrito se realiza con una tabla en PostgreSQL. A modo de ejemplo se toma el conjunto de datos *"breast_cancer.arff"*, provisto por Weka, con las modificaciones necesarias para su almacenamiento en PostgreSQL. Esta tabla tiene por nombre *breast_cancer*, cuenta con 286 filas y 11 columnas como se aprecia en la Figura 16.

	no. [PK] integer	age text	menopause text	tumor-size text	inv-nodes text	node-caps text	deg-malig text	breast text	breast-quad text	irradiat text	Class text
1	1	40-49	premeno	15-19	0-2	yes	3	right	left_up	no	recurrence-events
2	2	50-59	ge40	15-19	0-2	no	1	right	central	no	no-recurrence-events
3	3	50-59	ge40	35-39	0-2	no	2	left	left_low	no	recurrence-events
4	4	40-49	premeno	35-39	0-2	yes	3	right	left_low	yes	no-recurrence-events
5	5	40-49	premeno	30-34	3-5	yes	2	left	right_up	no	recurrence-events
6	6	50-59	premeno	25-29	3-5	no	2	right	left_up	yes	no-recurrence-events
7	7	50-59	ge40	40-44	0-2	no	3	left	left_up	no	no-recurrence-events
8	8	40-49	premeno	10-14	0-2	no	2	left	left_up	no	no-recurrence-events
9	9	40-49	premeno	0-4	0-2	no	2	right	right_low	no	no-recurrence-events
10	10	40-49	ge40	40-44	15-17	yes	2	right	left_up	yes	no-recurrence-events
11	11	50-59	premeno	25-29	0-2	no	2	left	left_low	no	no-recurrence-events
12	12	60-69	ge40	15-19	0-2	no	2	right	left_up	no	no-recurrence-events
13	13	50-59	ge40	30-34	0-2	no	1	right	central	no	no-recurrence-events
14	14	50-59	ge40	25-29	0-2	no	2	right	left_up	no	no-recurrence-events
15	15	40-49	premeno	25-29	0-2	no	2	left	left_low	yes	recurrence-events
16	16	30-39	premeno	20-24	0-2	no	3	left	central	no	no-recurrence-events
17	17	50-59	premeno	10-14	3-5	no	1	right	left_up	no	no-recurrence-events
18	18	60-69	ge40	15-19	0-2	no	2	right	left_up	no	no-recurrence-events
19	19	50-59	premeno	40-44	0-2	no	2	left	left_up	no	no-recurrence-events
20	20	50-59	ge40	20-24	0-2	no	3	left	left_up	no	no-recurrence-events
21	21	50-59	lt40	20-24	0-2		1	left	left_low	no	recurrence-events

Figura 16. Muestra de la tabla "breast_cancer" en PostgreSQL.

Siguiendo el proceso anterior se obtiene del agente Connector la tabla mostrada en la Figura 17. Luego se pasa al agente ARFF para editar el nombre de la relación (ver Figura 18), modificar los tipos de datos de los atributos a "nominal" (ver Figura 19), sin incluir la llave primaria de la tabla ya que no aporta información relevante para el archivo ARFF final. Finalmente se visualizan los datos antes de guardarlos (ver Figura 20), si el usuario desea puede ver una pre-visualización del archivo ARFF que se está construyendo (ver Figura 21).

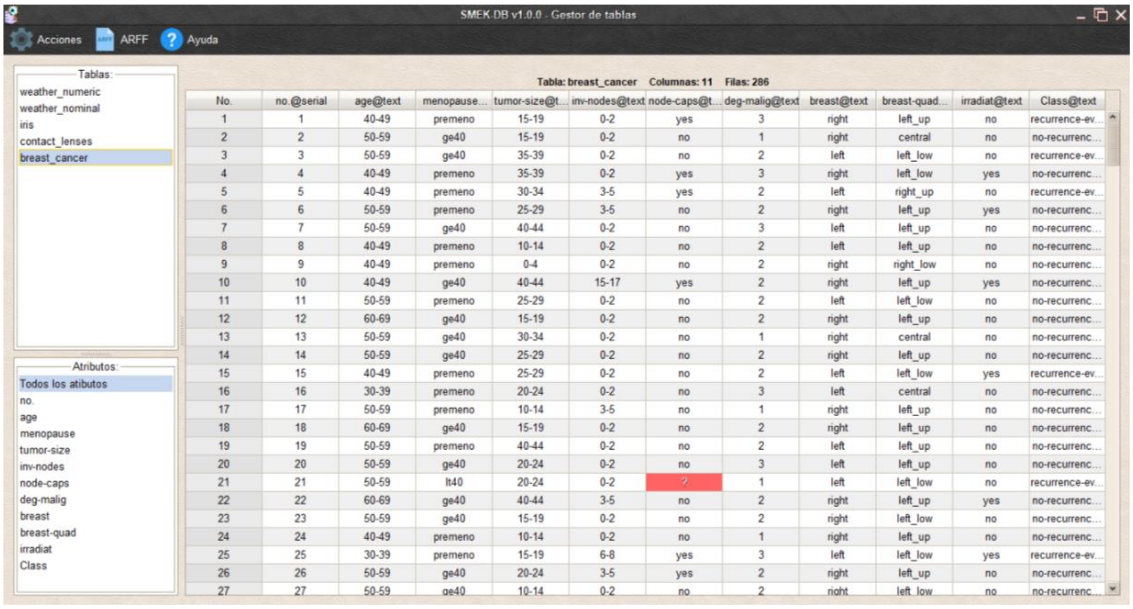

Figura 17. Interfaz de la tabla "breast_cancer". Agente Connector.

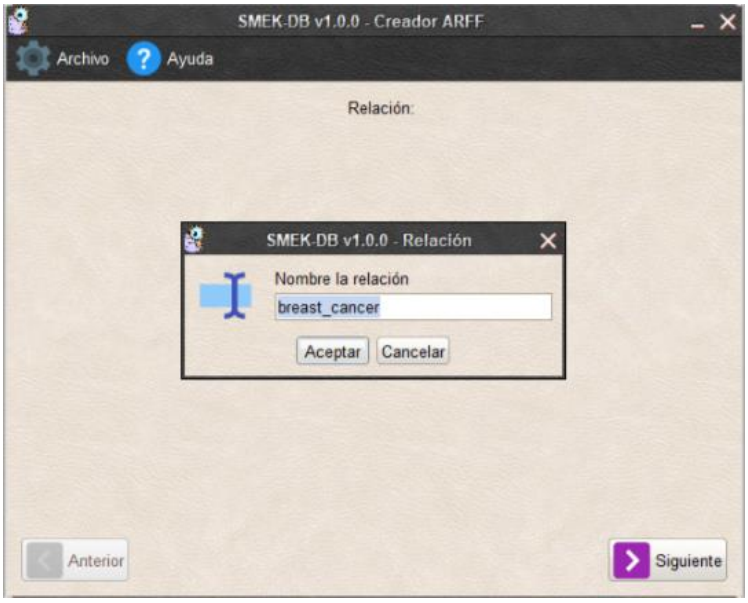

Figura 18. Interfaz para la edición del nombre de la relación. Agente Arff.

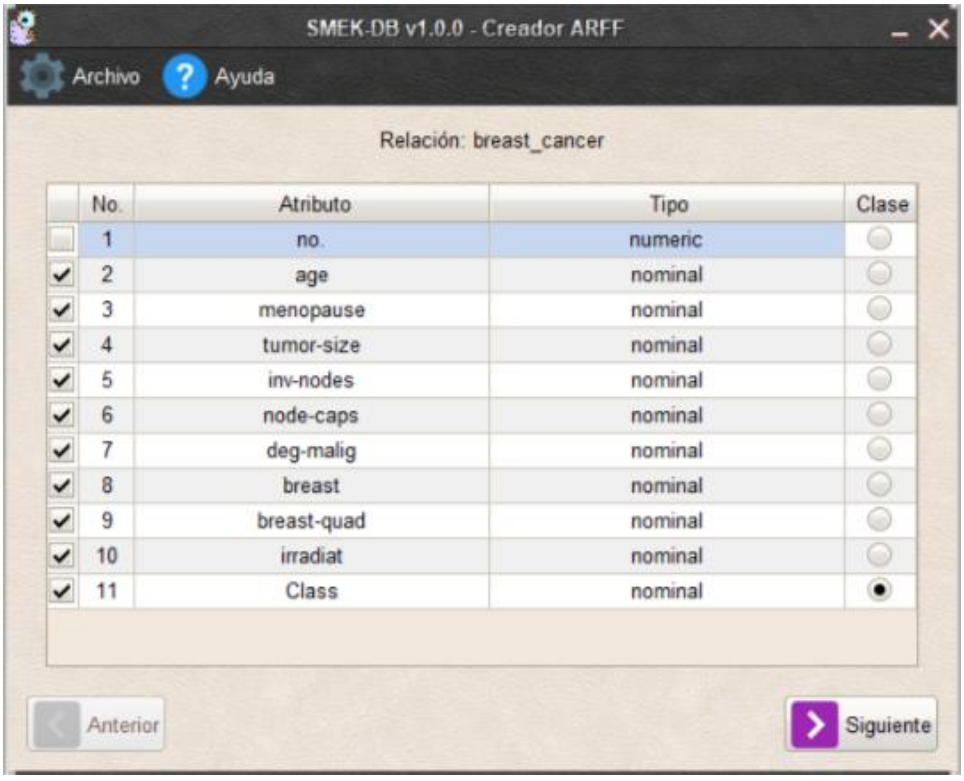

Figura 19. Interfaz para la selección y modificación de los atributos, la tabla "breast_cancer". Agente Arff.

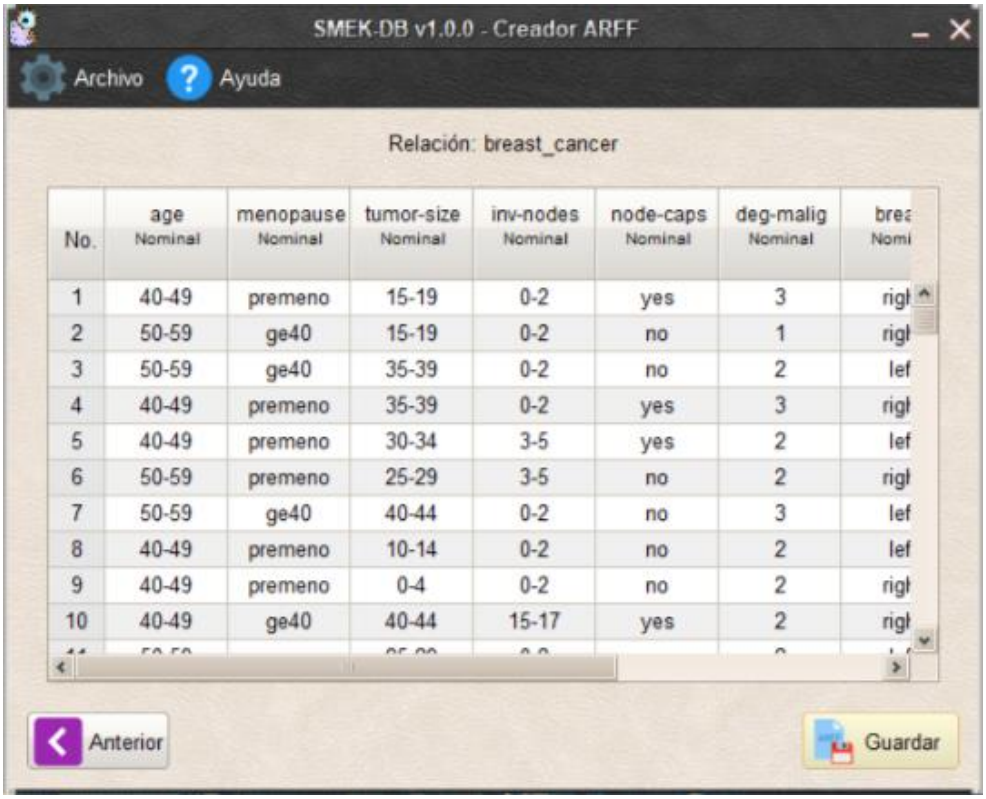

Figura 20. Interfaz para la edición de los datos, tabla "breast_cancer". Agente Arff.

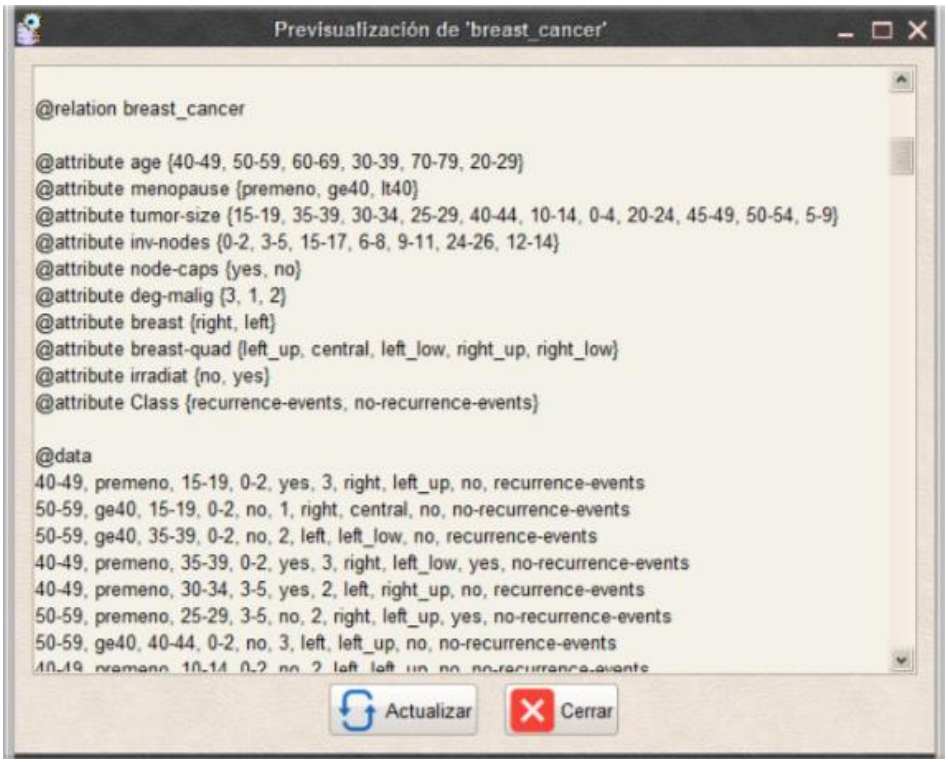

Figura 21. Interfaz de pre visualización del archivo ARFF obtenido de la tabla "breast_cancer". Agente Arff.

3.3 Resultados de la comparación de los clasificadores

En el proceso de comparación de clasificadores intervienen los agentes Smek y Classifier. Smek se encarga de recoger las configuraciones de los algoritmos, los métodos de evaluación y la dirección del archivo con los datos (ver Figura 22). El agente Classifier con la información recibida de Smek construye, evalúa y calcula las métricas para cada algoritmo. Luego compara los valores alcanzados en cada métrica para determinar y proponer el mejor algoritmo de clasificación y muestra los resultados en pantalla (ver Figura 23). Classifier también permite al usuario visualizar la comparación de las métricas en gráficos para una mejor comprensión visual (ver Figura 24).

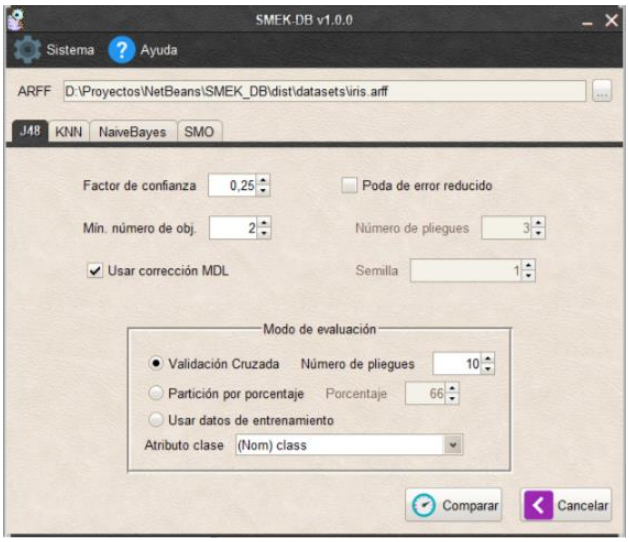

Figura 22. Interfaz de configuración de algoritmos de clasificación para su comparación.

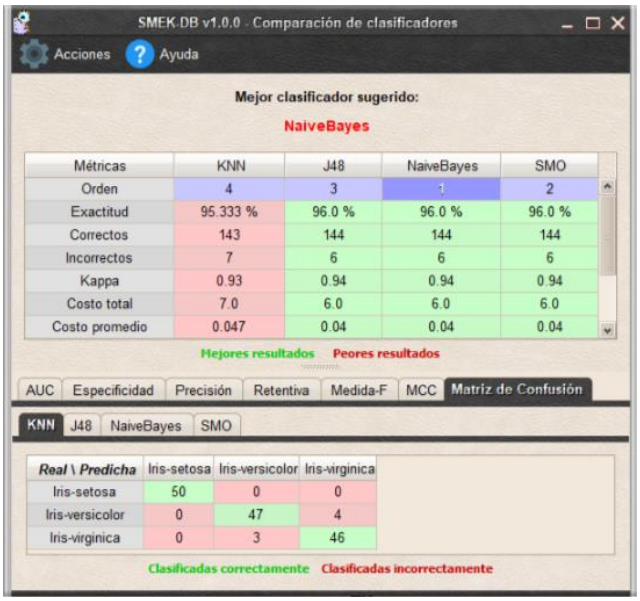

Figura 23. Interfaz de resultados de la comparación de algoritmos de clasificación.

Figura 24. Interfaz de visualización de resultados de la comparación de las métricas en clasificadores.

3.4 Resultados de la comparación de los agrupadores

En el proceso de comparación de agrupadores intervienen los agentes Smek y Clusterer. Smek se encarga de recoger las configuraciones de los algoritmos, los métodos de evaluación y la dirección del archivo con los datos (ver Figura 25). El agente Clusterer con la información recibida de Smek construye, evalúa y calcula las métricas para cada algoritmo. Luego compara los valores alcanzados en cada métrica para determinar y proponer el mejor algoritmo de agrupamiento mostrando los resultados en pantalla (ver Figura 26). Clusterer también permite al usuario visualizar la comparación de las métricas en gráficos para una mejor comprensión visual (ver Figura 27).

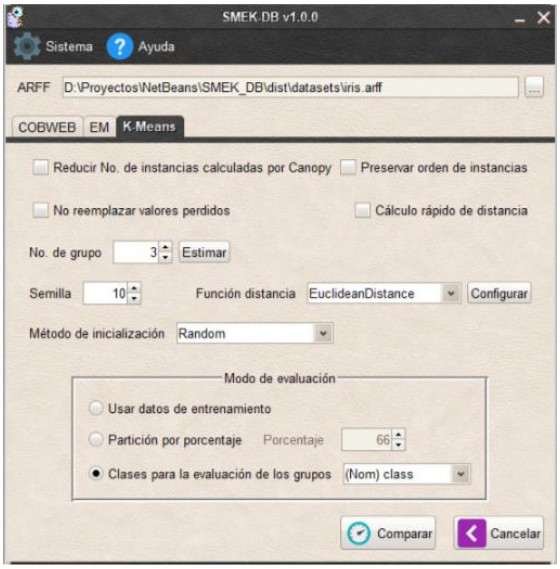

Figura 25. Interfaz de configuración de algoritmos de agrupamiento para su comparación.

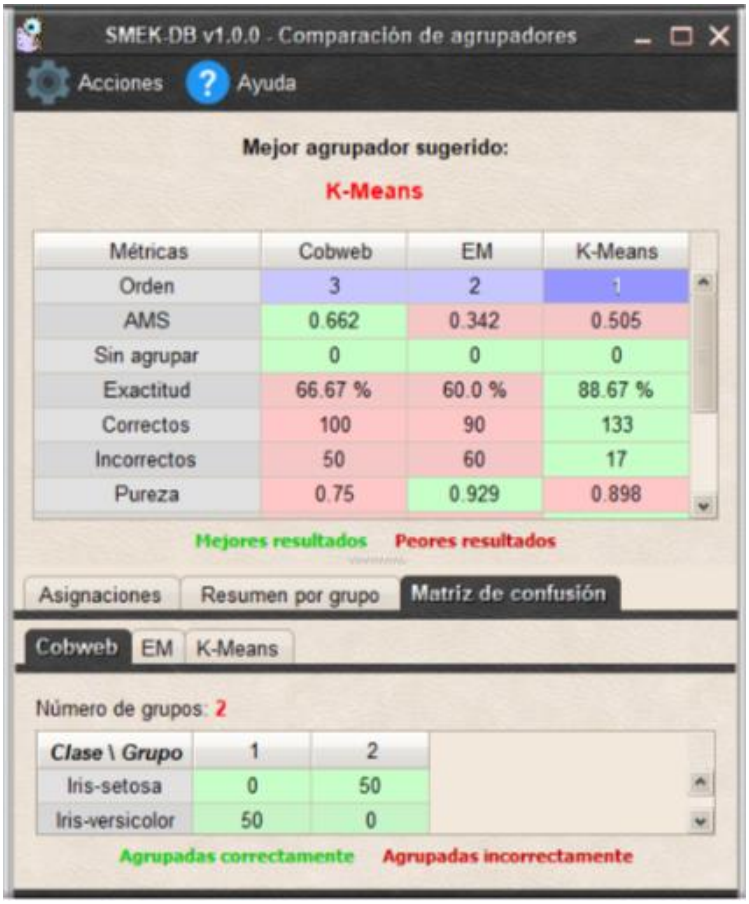

Figura 26. Interfaz de resultados de la comparación de algoritmos de agrupamiento.

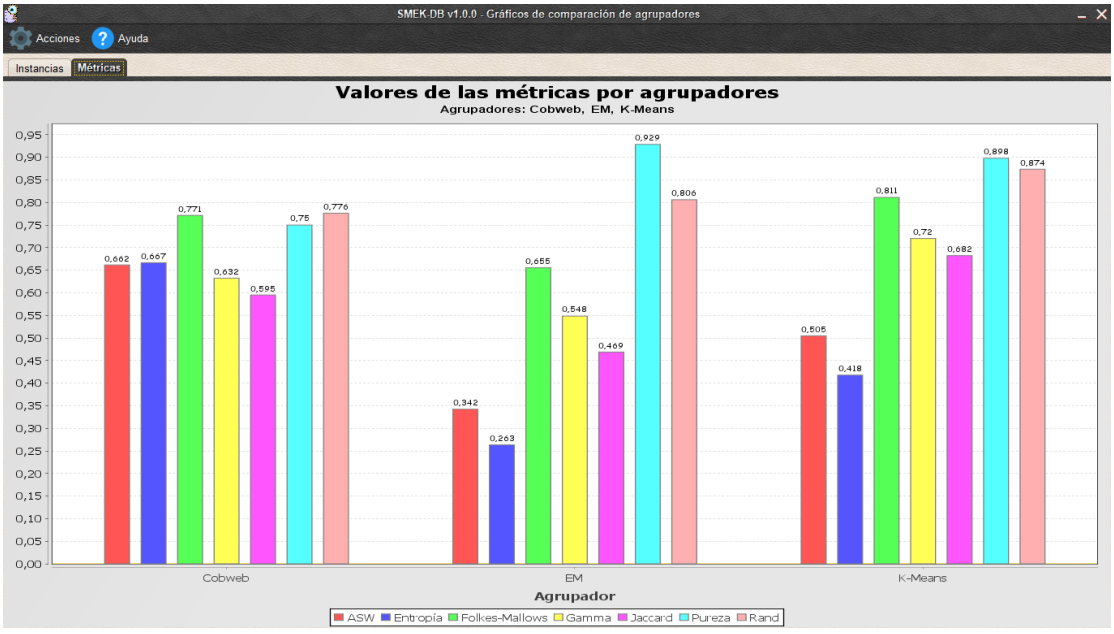

Figura 27. Interfaz de visualización de resultados de la comparación de las métricas en agrupadores.

3.5 Pruebas

Garantizar la calidad de un software se hace cada día más importante, debido al amplio margen de procesos a los que están encaminados. En este aspecto la ingeniería de software ofrece ciertos procedimientos que apoyan el fortalecimiento de la calidad. Como los sistemas multiagentes poseen características que los diferencian de la orientación a objetos, las metodologías destinadas a su desarrollo, al ser relativamente recientes no cuentan con una fase de prueba bien definida. En INGENIAS, según su autor, se puede llevar a acabo de igual manera que en un software convencional, sin embargo, esto no se cumple así porque el comportamiento de un SMA es emergente y por tanto difícil de pronosticar (46).

El V-Model es un modelo de procedimiento alemán para el desarrollo de software que prevé una secuencia fija de pasos de trabajo, divididos en una fase de desarrollo y una de comprobación. Los pasos individuales de la fase de desarrollo son paralelos en el contenido con los de la fase de comprobación, gráficamente se produce una V. La utilización del V-Model facilita la identificación de diferentes actividades y técnicas de pruebas, además, provee un marco de trabajo para revisar trabajos previos e identificar las necesidades de futuros encargos en algunas direcciones. En la Figura 28 se muestra el V-Model para la metodología INGENIAS, donde las fases del ciclo de vida del software se encuentran en la parte izquierda y las pruebas correspondientes con cada fase en la parte derecha (46).

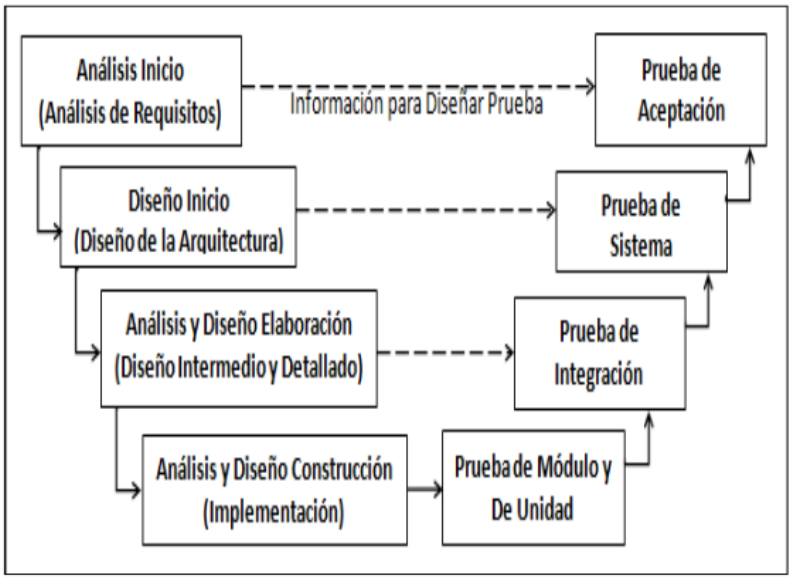

Figura 28. V-Model para la metodología INGENIAS. [fuente (46)]

La fase de análisis de requisitos es la encargada de recopilar las necesidades del usuario. Por lo que la Prueba de Aceptación se diseña para determinar si el software, una vez

completado, satisface estas necesidades, en otras palabras, se realiza para conocer si el software hace lo deseado por los usuarios. Esta prueba ha de involucrar a usuarios y otros individuos que tengan un fuerte conocimiento del negocio.

La fase de diseño de arquitectura en cualquier proceso de desarrollo de software es la guía para el desarrollo del mismo, en esta fase se diseñan los componentes de la aplicación lo cual permite visualizar la interacción de las entidades del negocio. De forma general, en esta fase se describe cómo se construirá la aplicación de software. En INGENIAS este eje de desarrollo lo constituye el meta-modelo de Organización, el cual se obtiene desde la primera fase de ejecución. Por lo tanto, el objetivo principal de la Prueba de Sistema en esta fase, se centra en el diseño de la arquitectura, y se valida el mapeo entre las especificaciones de los requisitos y el diseño. En la actividad de validar el diseño es importante chequear la correlación entre las metas del sistema y la capacidad y roles del agente (46).

Además, de las pruebas de aceptación y de sistema los autores proponen la realización de pruebas de rendimiento, destinadas a determinar el tiempo de respuesta del sistema ante determinadas características, con el objetivo de recomendar el trabajo con una cifra media de datos y ofrecer al usuario una experiencia satisfactoria.

3.5.1 Pruebas de Sistema

En un SMA las pruebas de sistemas están encaminadas a la validación del diseño y la verificación de la comunicación entre los distintos roles de los agentes. Siguiendo las actividades propuestas en (46), a continuación se presentan los diagramas de colaboración que describen los requisitos funcionales identificados.

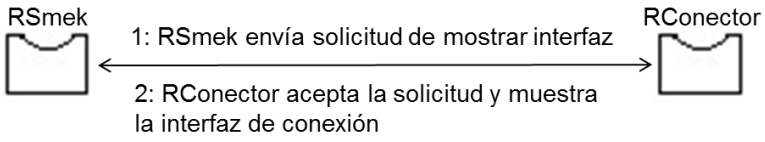

Figura 29. Diagrama de colaboración del RF1 Realizar conexión con PostgreSQL.

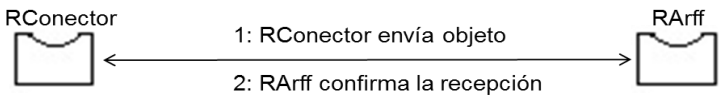

Figura 30. Diagrama de colaboración del RF2 Guardar tabla PostgreSQL como archivo ARFF.

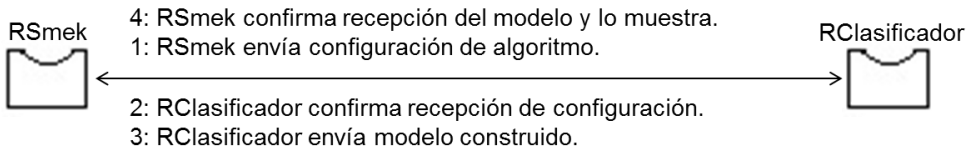

Figura 31. Diagrama de colaboración del RF3 Construir modelo de clasificación.

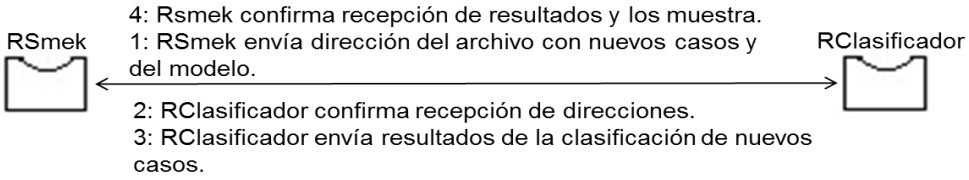

Figura 32. Diagrama de colaboración del RF4 Construir modelo de clasificación en paralelo.

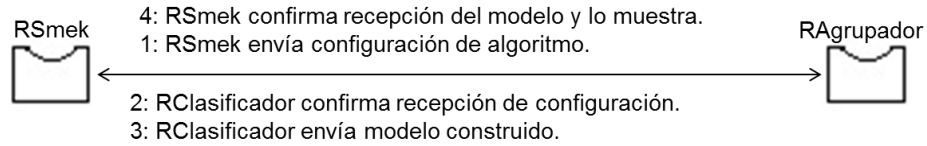

Figura 33. Diagrama de colaboración del RF5 Construir modelo de agrupamiento.

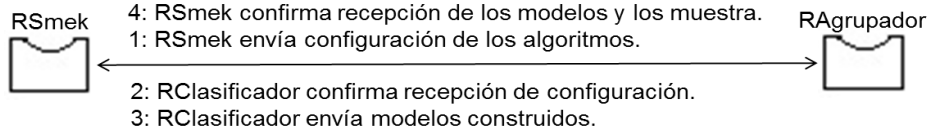

Figura 34. Diagrama de colaboración del RF6 Construir modelo de agrupamiento en paralelo.

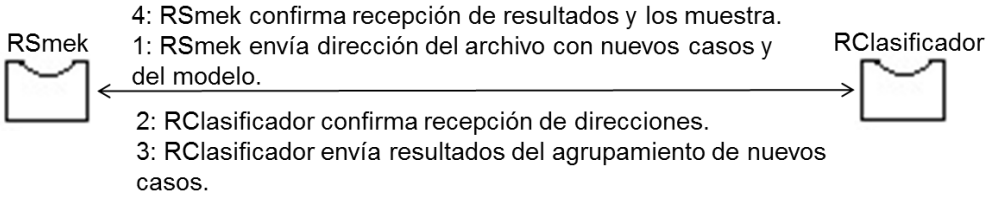

Figura 35. Diagrama de colaboración del RF7 Clasificar nuevos casos.

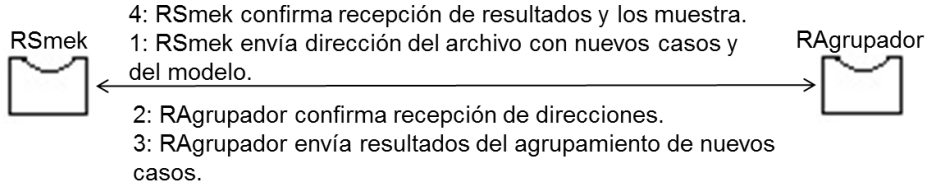

Figura 36. Diagrama de colaboración del RF8 Agrupar nuevos casos.

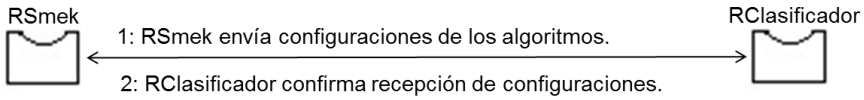

Figura 37. Diagrama de colaboración del RF9 Comparar algoritmos clasificadores.

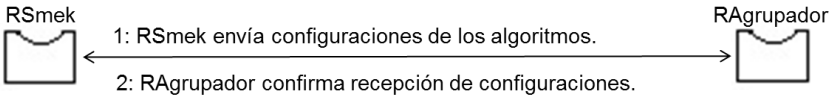

Figura 38. Diagrama de colaboración del RF10 Comparar algoritmos agrupadores.

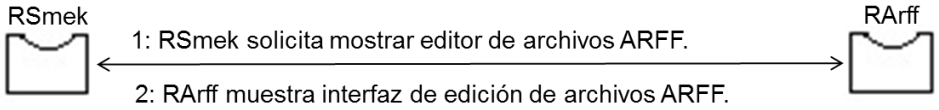

Figura 39. Diagrama de colaboración del RF12 Editar archivos ARFF.

Se realizaron un total de 11 pruebas de sistema, al Requisito Funcional 11 Visualizar modelos guardados, no se le aplicó una prueba de este tipo ya que no existen relaciones entre agentes, el agente Smek es el encargado de su ejecución. Estas pruebas validaron la correcta comunicación entre los distintos agentes del sistema.

3.5.2 Pruebas de Aceptación

Las pruebas de aceptación están enfocadas a evaluar si se consiguió la funcionalidad que los usuarios requerían. A continuación, se presentan las pruebas realizadas al sistema desarrollado.

Tabla 9. Caso de prueba de aceptación # 1.

Caso de prueba de aceptación	
Código: RF1	**Requisito Funcional:** 1
Nombre: Realizar conexión con PostgreSQL	
Descripción: Prueba si se realiza correctamente la conexión del sistema con PostgreSQL.	
Condición de ejecución: Debe estar disponible el servidor de bases de datos de PostgreSQL.	
Paso de ejecución: • Llenar los campos necesarios y seleccionar la opción “Actualizar”. • Elegir la base de datos. • Marcar la opción “Conectar” y luego “Continuar”.	
Resultados esperados: Se debe mostrar la interfaz que posibilita la creación del archivo ARFF.	
Evaluación de la prueba: Satisfactorio.	

Tabla 10. Caso de prueba de aceptación # 2.

Caso de prueba de aceptación	
Código: RF2	**Requisito Funcional:** 2
Nombre: Guardar tabla PostgreSQL como archivo ARFF	
Descripción: Prueba la funcionalidad guardar tabla de PostgreSQL como archivo ARFF.	
Condición de ejecución: Debe haberse realizado con éxito el Requisito Funcional 1.	
Paso de ejecución: • Seleccionar la tabla y luego la opción “Guardar ARFF”. • Escoger el nombre de la relación. • Especificar los atributos con sus tipos de datos. • Modificar algún dato si se desea y seleccionar la opción “Guardar”.	
Resultados esperados: Se debe guardar la tabla elegida como archivo ARFF.	
Evaluación de la prueba: Satisfactorio	

Tabla 11. Caso de prueba de aceptación # 3.

Caso de prueba de aceptación	
Código: RF3	**Requisito Funcional:** 3
Nombre: Construir modelo de clasificación	
Descripción: Prueba la funcionalidad construir modelo de clasificación.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar el algoritmo de clasificación que se desea aplicar. • Realizar las modificaciones deseadas en la configuración del algoritmo. • Seleccionar la opción “Iniciar”.	
Resultados esperados: Se debe mostrar una interfaz con el modelo del algoritmo de clasificación seleccionado.	
Evaluación de la prueba: Satisfactorio	

Tabla 12. Caso de prueba de aceptación # 4.

Caso de prueba de aceptación	
Código: RF4	**Requisito Funcional:** 4
Nombre: Construir modelo de clasificación en paralelo	
Descripción: Prueba la funcionalidad construir modelo de clasificación en paralelo.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar los algoritmos de clasificación que se desean aplicar.	

• Realizar las modificaciones deseadas en la configuración de los algoritmos. • Seleccionar la opción “Iniciar”.
Resultados esperados: Se debe mostrar una interfaz con los modelos de los algoritmos de clasificación seleccionados.
Evaluación de la prueba: Satisfactorio

Tabla 13. Caso de prueba de aceptación # 5.

Caso de prueba de aceptación	
Código: RF5	**Requisito Funcional:** 5
Nombre: Construir modelo de agrupamiento	
Descripción: Prueba la funcionalidad construir modelo de agrupamiento.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar el algoritmo de agrupamiento que se desea aplicar. • Realizar las modificaciones deseadas en la configuración del algoritmo. • Seleccionar la opción “Iniciar”.	
Resultados esperados: Se debe mostrar una interfaz con el modelo del algoritmo de agrupamiento seleccionado.	
Evaluación de la prueba: Satisfactorio	

Tabla 14. Caso de prueba de aceptación # 6.

Caso de prueba de aceptación	
Código: RF6	**Requisito Funcional:** 6
Nombre: Construir modelo de agrupamiento en paralelo	
Descripción: Prueba la funcionalidad construir modelo de agrupamiento en paralelo.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar los algoritmos de agrupamiento que se desean aplicar. • Realizar las modificaciones deseadas en la configuración de los algoritmos. • Seleccionar la opción “Iniciar”.	
Resultados esperados: Se debe mostrar una interfaz con los modelos de los algoritmos de agrupamiento seleccionados.	
Evaluación de la prueba: Satisfactorio	

Tabla 15. Caso de prueba de aceptación # 7.

Caso de prueba de aceptación	
Código: RF7	**Requisito Funcional:** 7
Nombre: Clasificar nuevos casos	
Descripción: Prueba la funcionalidad clasificar nuevos casos.	
Condición de ejecución: Se debe contar con un modelo obtenido como resultado de aplicar las funcionalidades de los requisitos Funcionales 3 o 4.	
Paso de ejecución: • Seleccionar el archivo ARFF con los nuevos casos y su modelo correspondiente. • Seleccionar la opción “Clasificar”.	
Resultados esperados: Debe mostrar en la interfaz los resultados de los nuevos casos clasificados.	
Evaluación de la prueba: Satisfactorio	

Tabla 16. Caso de prueba de aceptación # 8.

Caso de prueba de aceptación	
Código: RF8	**Requisito Funcional:** 8
Nombre: Agrupar nuevos casos	
Descripción: Prueba la funcionalidad agrupar nuevos casos.	
Condición de ejecución: Se debe contar con un modelo obtenido como resultado de aplicar las funcionalidades de los requisitos Funcionales 5 o 6.	
Paso de ejecución: • Seleccionar el archivo ARFF con los nuevos casos y su modelo correspondiente. • Seleccionar la opción “Agrupar”.	
Resultados esperados: Debe mostrar en la interfaz los resultados de los nuevos casos agrupados.	
Evaluación de la prueba: Satisfactorio	

Tabla 17. Caso de prueba de aceptación # 9.

Caso de prueba de aceptación	
Código: RF9	**Requisito Funcional:** 9
Nombre: Comparar algoritmos clasificadores	
Descripción: Prueba la funcionalidad comparar algoritmos clasificadores.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar la opción “Comparar Clasificadores”. • Configurar, si se desea, los algoritmos clasificadores. • Seleccionar la opción “Comparar”.	

Resultados esperados: Se debe mostrar una interfaz con los resultados de la comparación.
Evaluación de la prueba: Satisfactorio

Tabla 18. Caso de prueba de aceptación # 10.

Caso de prueba de aceptación	
Código: RF10	**Requisito Funcional:** 10
Nombre: Comparar algoritmos agrupadores	
Descripción: Prueba la funcionalidad comparar algoritmos agrupadores.	
Condición de ejecución: Se necesita un archivo ARFF (guardado previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar la opción “Comparar Agrupadores”. • Configurar, si se desea, los algoritmos agrupadores. • Seleccionar la opción “Comparar”.	
Resultados esperados: Se debe mostrar una interfaz con los resultados de la comparación.	
Evaluación de la prueba: Satisfactorio	

Tabla 19. Caso de prueba de aceptación # 11.

Caso de prueba de aceptación	
Código: RF11	**Requisito Funcional:** 11
Nombre: Visualizar modelos guardados	
Descripción: Prueba la funcionalidad de la visualización de los modelos guardados.	
Condición de ejecución: Se necesita tener modelos construidos previamente como resultados de los Requisitos Funcionales 3, 4, 5 o 6.	
Paso de ejecución: • Seleccionar el (los) modelos que se desean visualizar.	
Resultados esperados: Una interfaz que permita visualizar los modelos seleccionados.	
Evaluación de la prueba: Satisfactorio	

Tabla 20. Caso de prueba de aceptación # 12.

Caso de prueba de aceptación	
Código: RF12	**Requisito Funcional:** 12
Nombre: Editar archivos ARFF	
Descripción: Prueba la funcionalidad editar archivos ARFF.	
Condición de ejecución: Se necesita uno o varios archivos ARFF (guardados previamente o como resultado del Requisito Funcional 2)	
Paso de ejecución: • Seleccionar el (los) archivos ARFF.	

• Modificar, si se desea, los datos. • Guardar los cambios realizados.
Resultados esperados: Una interfaz que posibilite la edición de los archivos ARFF escogidos.
Evaluación de la prueba: Satisfactorio

Se realizaron 12 pruebas de aceptación, una por cada requisito funcional. Estas pruebas arrojaron como resultado, 12 casos “Satisfactorios” y ningún caso “Insatisfactorio”.

3.5.3 Pruebas de Rendimiento

La realización de las pruebas de rendimiento se basó en la selección de archivos ARFF con distintas cantidades de instancias y atributos. A estos se le aplicaron las funciones de comparación de algoritmos de clasificación y agrupamiento. Se verificaron los tiempos de respuesta de la aplicación, desde el inicio del proceso hasta que se sugiere el mejor algoritmo. Se tuvo en cuenta los requisitos mínimos de hardware y se aplicaron las pruebas en distintas computadoras. Los resultados de estas pruebas se observan en la Tabla 21, donde se puede apreciar que los tiempos de los clasificadores son relativamente pequeños. Sin embargo, no sucede la misma situación con los agrupadores, donde se obtienen tiempos largos a medida que aumentan las instancias y los atributos.

Tabla 21. Resultados de las pruebas de rendimiento.

No. de instancias	No. de atributos	Comparación de clasificadores	Comparación de agrupadores
100	10	0.83s	0.72s
	50	0.89s	1.31s
	100	0.94s	2.06s
1 000	10	1.74s	1min 39s
	50	9.27s	4min 33.6s
	100	17.72s	4min 54s
10 000	10	47.63s	46min 19.8s
	50	5min 0.09s	Más de 50min
	100	19min 3s	Más de 1h

3.6 Posibles aplicaciones del sistema

El sistema multiagente propuesto puede ser utilizado durante la toma de decisiones en problemas de genética para la clasificación de cadenas de ADN, pronósticos del

clima y el tráfico, en la selección de clientes potenciales, entre otros. Por ejemplo, si se desea desarrollar un software que ayude en la realización de diagnósticos médicos y se desconoce cuál algoritmo es apropiado emplear para minimizar las predicciones incorrectas; la propuesta de solución sugiere cual algoritmo utilizar basándose en los resultados obtenidos con los datos de entrenamiento para dicho problema.

3.7 Consideraciones finales del Capítulo

Como parte del desarrollo del sistema se expuso de forma detallada la implementación del proceso de transformación de las bases de datos de PostgreSQL a ficheros ARFF según los objetivos de cada uno de los agentes que intervienen. A partir de un estándar de codificación definido para la implementación se expusieron ejemplos de la comparación de los algoritmos. Por último, a partir de las pruebas realizadas, tanto las internas como de aceptación y rendimiento, se evidenció un funcionamiento satisfactorio del sistema.

CONCLUSIONES

La realización de este trabajo permitió arribar a las siguientes conclusiones:

- Dentro de las metodologías de desarrollo orientadas a agentes, INGENIAS agrupa las características necesarias para el desarrollo del SMA propuesto.
- Las no conformidades de conexión de Weka con PostgreSQL fueron solventadas con la implementación de la funcionalidad de transformación de bases de datos a archivos ARFF.
- Las métricas de análisis de algoritmos de clasificación, definidas en Weka, permiten compararlos para determinar el que alcanza mejores resultados al aplicarlos a un conjunto de datos específicos.
- A partir de la implementación de métricas, identificadas en la bibliografía consultada, fue posible la comparación de algoritmos de agrupamiento para determinar cuál alcanza mejores resultados al aplicarlo a un conjunto de datos específicos.
- La utilización de la metodología INGENIAS y las herramientas seleccionadas, permitió desarrollar un sistema multiagente capaz de identificar de forma automática el mejor algoritmo de clasificación o agrupamiento a aplicar a una base de datos en PostgreSQL o a ficheros ARFF de Weka, atendiendo a un conjunto de métricas.

RECOMENDACIONES

Se proponen las siguientes recomendaciones con el objetivo de mejorar el sistema desarrollado:

- Incorporar al sistema otros algoritmos de clasificación o agrupamiento implementados por Weka.
- Ampliar el trabajo con otros formatos de archivos de datos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Russell, Stuart y Norvig, Peter.** *Inteligencia Artificial. Un enfoque moderno.* 2da. 2008. pág. 1.

2. *Integración de Minería de Datos y Sistemas Multiagentes: un campo de investigación y desarrollo.* **Molero Castillo, Guillermo y Meda Campaña, María Elena.** 3, La Habana, Cuba : s.n., septiembre-diciembre de 2010, Ciencias de la Información, Vol. 41, págs. 53-56. ISSN.

3. **Valdés Rabelo, Sergio Alberto y Godales Quiñones, Sadiel.** *Propuesta de arquitectura para sistemas de agentes inteligentes en el CESIM.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2012. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

4. **García Herrero, Jesús y Molina López, José Manuel.** *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 1-5.

5. —. *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 96-98.

6. **García Herrero, Jesús y Molina López, José Manuel.** *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 120-122.

7. *Statistical Comparisons of the Top 10 Algorithms in Data Mining for Classification Task.* **Settouti, Nesma, Bechar, Mohammed El Amine y Amine Chikh, Mohammed.** 1, 2016, International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 4.

8. **García Herrero, Jesús y Molina López, José Manuel.** *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 123-143.

9. *Book Review: C4.5: Programs for Machine Learning.* **Quinlan, John Ross.** Boston : Kluwer Academic Publishers, 1994.

10. *Estimating continuous distributions in bayesian classifiers.* **John, George H. y Langley, Pat.** California : Morgan Kaufmann Publishers, 1995.

11. **Candel Contardo, Diego.** Tesis para optar por el grado académico de magíster en Ciencias de la Ingeniería Informática. *Algoritmo tipo SMO para la AD-SVM aplicado a la clasificación multicategoría.* 2011. pág. 5.

12. *Fast Training of Support Vector Machines Using.* **Platt, John C.** Febrero de 1999.

13. **García Herrero, Jesús y Molina López, José Manuel.** *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 98-106.

14. **Michalski, R. S., Stepp, J. G. y Mitchell, T. M.** *Learning from observation: Conceptual clustering.* s.l. : Machine Learning: An Artificial Intelligence Approach, 1983.

15. **Garre, Miguel, Cuadrado, Juan José y Silicia, Miguel Ángel.** *Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software.* Madrid : s.n. págs. 3-4.

16. **Ingaramo, Diego A., Errecalde, Marcelo L. y Rosso, Paolo.** Medidas internas y externas en el agrupamiento de resúmenes científicos de dominios reducidos. España : s.n., 2007. 39, págs. 56-59.

17. **García Herrero, Jesús y Molina López , José Manuel.** *Técnicas de análisis de datos.* Madrid : s.n., 2012. págs. 104-106.

18. **Russell, Stuart y Norvig, Peter.** *AI. A Modern Approach.* 3ra. 2010.

19. —. *Inteligencia Artificial. Un enfoque moderno.* 2da. 2008. págs. 37-39.

20. **ECSDI.** *Sistemas Multiagente.* curso 2015-2016.

21. **Llamas Bello, César.** *Introducción a los Agentes y Sistemas Multiagente.* 2000.

22. **Marchetti, Tulio José y García, Alejandro Javier.** *Plataformas para Desarrollo de Sistemas Multiagente. Un análisis comparativo.*

23. **FIPA.** ***Foundation for Intelligent Physical Agents.*** [En línea] http://www.fipa.org.

24. JADE. [En línea] http://jade.tilab.com.

25. IEEE Spectrum. [En línea] http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages.

26. Netbeans. [En línea] http://www.netbeans.org.

27. Hernández Orallo, José y Ferri Ramírez, César. Práctica de Minería de Datos. Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería de Software. 2006.

28. PostgreSQL-es. [En línea] http://www.postgresql.org.

29. DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. [En línea] https://db-engines.com/en/ranking.

30. Gómez Sanz, Jorge J. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. pág. 27.

31. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. pág. 30.

32. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 39-46.

33. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 62-81.

34. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 82-87.

35. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 95-96.

36. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 115-117.

37. —. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 123-127.

38. Corso, Cynthia Lorena. *Aplicación de algoritmos de clasificación supervisada usando Weka.* Universidad Tecnológica Nacional. Córdoba : s.n., 2009.

39. Powers, David Martin. *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation.* School of Informatics and Engineering Flinders University. Adelaide : s.n., 2011. Technical Report.

40. *Understanding Interobserver Agreement: The Kappa Statistic.* **Viera, Anthony J. y Garret, Joanne M.** 5, mayo de 2005, Family Medicine, Vol. 37, pág. 361.

41. Fawcett, Tom. *An introduction to ROC analysis.* Institute for the Study of Learning and Expertise. Palo Alto : s.n., 2005.

42. Tan, Pang-Ning, Steinbach, Michael y Kumar, Vipin . *Introduction to Data Mining.* 2006. págs. 541-552.

43. Gómez Sanz, Jorge J. *Memoria que presenta para optar al grado de Doctor. Modelado de Sistemas Multi-Agentes.* Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid. Madrid : s.n., 2003. págs. 123-127.

44. Sommerville, Ian. *Software Engineering.* 9th Edition. 2011.

45. Newell, A. The Knowledge level, Artificial Intelligence. 1982. Vol. 18, págs. 87-127.

46. *Propuesta de actividades de pruebas para Ingenias.* **Hadfeg Fernández, Yahima, Moreno Espino, Mailyn y Delgado Dapena, Martha Dunia.** No. 3, septiembre-diciembre de 2012, Revista Cubana de Ingeniería, Vol. III, págs. 47-56. ISSN 2223 -1781.