

*Universidad de las Ciencias Informáticas*

*Facultad 5*



*Opciones de visualización de los algoritmos de análisis de datos estudiados en*

*Inteligencia Artificial 2 en el software de Minería de datos*

*WELA*

*Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas*

*Autores:*

*Hayron Yudiel Pérez Gómez*

*Handy Pereira Arencibia*

*Tutores:*

*Ing. Joan Larra Nápoles*

*Ing. Elizabeth Enriquez Guisado*

*Ing. José Ricardo García Arozarena*

*Junio, 2015*

*"Año 57 de la Revolución"*

**DECLARACIÓN DE AUTORÍA**

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los \_\_\_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

---

Hayron Yudiel Pérez Gómez

---

Handy Pereira Arencibia

---

Ing. Yoan Parra Nápoles

---

Ing. Elizabeth Enríquez Guisado

---

Ing. José Ricardo García Arozarena

## **AGRADECIMIENTOS**

*De Cayron*

*Después de muchos años de estudio y esfuerzo, hoy me gradúo de Ingeniero en Ciencias Informáticas. Para hacer realidad este sueño muchas personas me ayudaron, apoyaron y depositaron su confianza en mí. Hoy tengo la oportunidad de hacerles saber cuan agradecido estoy.*

*A mis padres por todo su esfuerzo y sacrificio, por su cariño, su apoyo incondicional, gracias por estar siempre cuando los necesité. Quiero que sepan que los quiero y los admiro mucho y este logro es de ustedes también, nunca le podré estar lo suficientemente agradecido por todo lo que han hecho por mí, gracias por ser mi guía, mi apoyo y mi vida. A mi hermano por todo lo que hemos vivido juntos y por estar siempre a mi lado cuando lo necesité, siempre ha sido un ejemplo a seguir, trabajador, buen hermano, mi mejor amigo, incluso hasta mi padre, solo quiero que sepa que lo quiero con la vida y también por tener otra de las personas por la cual sería capaz de dar mi vida si lo necesitara, que aunque no sea mi hija la quiero como tal, quiero que sepas que eres mi niña Naye.*

*A mi abuela por todo su cariño, a mis primos Deisy y Lazarito por ser como mis hermanos, a mi tío Lázaro por aconsejarme en todo momento y ser otro padre para mí, a mi tía Mary por todo el apoyo que siempre nos dio a mi hermano y a mí cuando lo necesitamos, a Analeiza, a mi cuñada Rosaly y a Isabel que de alguna forma han estado en mi vida y han influido en mí para ser una mejor persona, en general a toda mi familia por el cariño y el apoyo que me han dado.*

*Por otra parte quisiera agradecerles a mis tutores Parra, Elizabeth, Ricardo y Luisito por todo su esfuerzo y paciencia, son una de las piezas más importantes en esta tesis y más que tutores y profesores son mis amigos y quiero que sepan que pueden contar conmigo para lo que sea.*

*A mi compañero de tesis Handy por todas las malas noches que hemos pasado durante la realización de esta tarea tan importante en nuestra vida, por ser mi amigo, mi hermano y por apoyarme en todo momento que hemos vivido juntos y también agradecerle a toda su familia.*

*También a las personas que forman el día a día de mi vida junto a mi familia, a Lidia, por todo el apoyo que me ha brindado, por soportarme y aconsejarme en los momentos buenos y malos, por ser como otra madre para mí, a mi piquete, tanto los que están fuera de la escuela como los que están aquí, el Niño, Aniel, Armando, Javielito, Eliades, Luisito, Edward, Yalbert, el Jimmy, Rey, el Droga (Yamir), Raúl, el Hobbit (Orlando), el Zurdo, Eric y Maiyara los cuales hemos vivido grandes momentos juntos, nos hemos apoyado los unos a los otros y más que mis amigos los considero a todos mis hermanos y los quiero muchísimo.*

*A mi novia Leidy por ser lo más lindo que me ha pasado en este curso, por soportarme y quererme, por apoyarme y darme fuerzas para seguir adelante solo quiero que sepas que te amo con la vida mi princesa.*

*A todos aquellos que de una forma u otra han formado parte de mi vida y me han apoyado en todo momento, a la pura Lisandra, Diomne, Beatriz, Rachel, Yaike, José Raúl, Emilio, el loco de Frank, Leovanis, el Chiqui, al piquete del fútbol, quiero que sepan que pueden contar conmigo para lo que sea que aquí tienen un amigo más.*

*En fin a todos aquellos que de una forma u otra han formado parte de mi vida y me han apoyado gracias a todos.*

*De Candy*

*Después de muchos años de estudio y esfuerzo, hoy me gradúo de Ingeniero en Ciencias Informáticas. Para hacer realidad este sueño muchas personas me ayudaron, apoyaron y depositaron su confianza en mí. Hoy tengo la oportunidad de hacerles saber cuan agradecido estoy.*

*A mis padres por todo su esfuerzo y sacrificio, por su cariño, su apoyo incondicional, gracias por estar siempre cuando los necesité. Quiero que sepan que los quiero y los admiro mucho y este logro es de ustedes también, nunca le podré estar lo suficientemente agradecido por todo lo que han hecho por mí, gracias por ser mi guía, mi apoyo y mi vida. A mi hermano por todo lo que hemos vivido juntos y por estar siempre a mi lado cuando lo necesité, siempre ha sido un ejemplo a seguir, trabajador, buen hermano, mi mejor amigo, incluso hasta mi padre, solo quiero que sepa que lo quiero con la vida y también por tener otra de las personas por la cual sería capaz de dar mi vida si lo necesitara, que aunque no sea mi hijo lo quiero como tal, a mi cuñada que de alguna forma ha estado en mi vida y ha influido en mí, para ser una mejor persona. En general a toda mi familia por el cariño y el apoyo que me han dado.*

*Por otra parte quería agradecerles a mis tutores Parra, Elizabeth, Ricardo y Luisito por todo su esfuerzo y paciencia, son una de las piezas más importantes en esta tesis y más que tutores y profesores son mis amigos y quiero que sepan que pueden contar conmigo para lo que sea. Además de agradecerle a la jefa de 5to año Susej.*

*A mi compañero de tesis Hayron por todas las malas noches que hemos pasado durante la realización de esta tarea tan importante en nuestra vida, por ser mi amigo, mi hermano y por apoyarme en todo momento que hemos vivido juntos y también agradecerle a toda su familia.*

*También a las personas que forman el día a día de mi vida junto a mi familia, a Lidia, por todo el apoyo que me ha brindado, a mi piquete tanto los que están fuera de la escuela como los que están aquí, el Nino, Anniel, Armando, Eliades, Luisito, Edward, Yalbert, el Jimmy, Raúl, el Hobbit (Orlando), al Zurdo, a Vicente los cuales hemos vivido grandes*

*momentos juntos, nos hemos apoyado los unos a los otros y más que mis amigos los considero a todos mis hermanos y los quiero muchísimo.*

*A todos aquellos que de una forma u otra han formado parte de mi vida y me han apoyado en todo momento, a la pura Lisandra, Diomne, Beatriz, Rachel, Yaiquel, José Raúl, Emilio, el loco de Frank, Leovanis, al Yuli, al Persa, Maiyara, a Lindamelia, Ingrid, Alí, a Erick, Jiubel, Aníbal, Gedry, Lázaro, Anabel, el piquete del fútbol, a mi gente de la facultad de Artemisa no los olvido, a los de la facultad 7, a mi aula, a los de la facultad 2, a mis amigos de la FICI, a los Alejandro, sepan que pueden contar conmigo para lo que sea que aquí tienen un amigo más.*

*En fin a todos aquellos que de una forma u otra han formado parte de mi vida y me han apoyado gracias a todos.*

## DEDICATORIA

*De Cayron*

*A mis padres, mi hermano, mi sobrina y a todas las personas que me quieren y siempre confiaron en mí brindándome su apoyo para que viera realizado mi sueño, quiero regalarles este momento y honrarlos por tanto amor y dedicación.*

*Los quiero mucho.*

*De Candy*

*A mi familia, en especial a mis padres, mi hermano y a todas las personas que me quieren y siempre confiaron en mí brindándome su apoyo para poder realizar este sueño, quiero regalarles este momento y honrarlos por tanto amor y dedicación. Los*

*quiero mucho.*

## **RESUMEN**

WEKA (del inglés *Waikato Environment for Knowledge Analysis*) es un software que consiste en un conjunto de algoritmos de aprendizaje para tareas de minería de datos (MD). Se trata de una herramienta totalmente desarrollada bajo la Licencia Pública General en el lenguaje de programación Java, que contiene una interfaz gráfica para la aplicación de los algoritmos directamente al conjunto de datos de origen. En la Universidad de las Ciencias Informáticas, específicamente en el Tema 2, Aprendizaje Automático de la asignatura de Inteligencia Artificial 2 (IA2) se estudian las diferentes fases del proceso de MD y se hace uso del software WEKA para impartir esta asignatura. Sin embargo esta herramienta no se encuentra orientada a la comprensión del funcionamiento interno de los algoritmos que se estudian en clases como: los Árboles de Decisión, K-Medias, K-Vecino más cercano y Redes Neuronales. En el presente trabajo se desarrollan nuevas opciones de visualización para los algoritmos antes mencionados en el software WEKA, lo cual le permitirá a los profesores y estudiantes disponer del software como apoyo para entender mediante gráficas y nuevos esquemas de visualización la MD, así como el preprocesamiento de los datos a través de los algoritmos estudiados en las clases.

**Palabras Claves:** visualización, WEKA, árboles de decisión, Redes Neuronales, K-Medias, K-Vecino más cercano.



**ÍNDICE DE CONTENIDOS**

<b>DECLARACIÓN DE AUTORÍA</b> .....	II
<b>AGRADECIMIENTOS</b> .....	III
<b>DEDICATORIA</b> .....	VII
<b>RESUMEN</b> .....	VIII
<b>ÍNDICE DE CONTENIDOS</b> .....	IX
<b>ÍNDICE DE FIGURAS</b> .....	XII
<b>ÍNDICE DE TABLAS</b> .....	XIII
<b>INTRODUCCIÓN</b> .....	1
<b>CAPÍTULO 1: FUNDAMENTOS TEÓRICOS</b> .....	4
<b>1.1 Minería de datos</b> .....	4
<b>1.1.1 Áreas en las que se utiliza la MD</b> .....	4
<b>1.1.2 Fases de la MD</b> .....	5
<b>1.2 Inteligencia Artificial</b> .....	6
<b>1.3 Algoritmos de MD</b> .....	7
<b>1.3.1 Árbol de Decisión (AD)</b> .....	7
<b>1.3.2 K-Means</b> .....	8
<b>1.3.3 K-Vecino más cercano</b> .....	8
<b>1.3.4 Redes neuronales</b> .....	9
<b>1.4 Herramientas de MD</b> .....	10
<b>1.4.1 WEKA</b> .....	10
<b>1.4.2 Desventajas de la visualización de los algoritmos en el software de MD WEKA</b> .....	11
<b>1.5 Lenguaje de programación</b> .....	11
<b>1.5.1 Java</b> .....	12
<b>1.6 Entorno de Desarrollo Integrado (IDE)</b> .....	12
<b>1.6.1 Eclipse</b> .....	12
<b>1.6.2 NetBeans</b> .....	13
<b>1.6.3 Fundamentación del IDE seleccionado</b> .....	13
<b>1.7 Herramienta de modelado</b> .....	14
<b>1.7.1 Visual Paradigm</b> .....	14
<b>1.7.2 Enterprise Architect</b> .....	15
<b>1.7.3 Rational Rose Enterprise Edition</b> .....	16
<b>1.7.4 Fundamentación de la herramienta de modelado seleccionada</b> .....	16
<b>1.8 Metodología de desarrollo de software</b> .....	17
<b>1.8.1 Programación Extrema</b> .....	17

1.8.2	Proceso Unificado de Desarrollo .....	17
1.8.3	Fundamentación de la metodología seleccionada.....	18
1.9	Patrones de diseño .....	18
1.9.1	Patrones de asignación de responsabilidades o GRASP .....	19
1.9.2	Patrones GoF.....	20
1.10	Conclusiones parciales .....	20
<b>CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN.....</b>		<b>21</b>
2.1	Propuesta de solución .....	21
2.2	Diagrama de paquetes .....	21
2.3	Requisitos del sistema.....	22
2.3.1	Requisitos funcionales .....	22
2.3.2	Requisitos no funcionales .....	23
2.3.3	Lista de reserva del producto .....	24
2.4	Historia de usuario.....	25
2.5	Fase de planificación .....	26
2.5.1	Estimación de esfuerzo .....	26
2.5.2	Plan de iteraciones .....	27
2.5.3	Plan de entrega .....	27
2.6	Fase de diseño .....	28
2.6.1	Arquitectura de diseño .....	28
2.6.2	Patrones GRASP utilizados .....	28
2.6.3	Patrones GoF utilizados .....	29
2.6.4	Tarjetas de Clase-Responsabilidad-Colaboración .....	29
2.7	Conclusiones parciales .....	30
<b>CAPÍTULO 3: DESCRIPCIÓN DE LOS RESULTADOS .....</b>		<b>31</b>
3.1	Estándar de codificación.....	31
3.2	Fase de implementación .....	31
3.3	Integración de las opciones de visualización a otras versiones de WEKA 32	
3.4	Fase de prueba .....	33
3.4.1	Pruebas unitarias .....	33
3.4.2	Pruebas de aceptación .....	36
3.5	Conclusiones parciales .....	37
<b>CONCLUSIONES GENERALES.....</b>		<b>38</b>
<b>RECOMENDACIONES .....</b>		<b>39</b>
<b>BIBLIOGRAFÍA.....</b>		<b>40</b>

<b>GLOSARIO DE TÉRMINOS</b> .....	44
<b>ANEXOS</b> .....	45

**ÍNDICE DE FIGURAS**

Figura 1. Ejemplo de donde sustituir las clases. .... 33  
Figura 2. 1ra Instancia de la Prueba unitaria. .... 34  
Figura 3. 2da Instancia de la Prueba unitaria. .... 35  
Figura 4. 3ra Instancia de la Prueba unitaria. .... 35  
Figura 5. 4ta Instancia de la Prueba unitaria. .... 36  
Figura 6. Patrones GoF. .... 45  
Figura 7. Diagrama de paquetes. .... 45  
Figura 8. Acta de validación. .... 62

**ÍNDICE DE TABLAS**

Tabla 1. Requisitos funcionales. ....	23
Tabla 2. Lista de reserva del producto RF. ....	24
Tabla 3. Lista de reserva del producto RNF. ....	25
Tabla 4. Historia de Usuario HU1. ....	26
Tabla 5. Plan de esfuerzo por HU. ....	26
Tabla 6. Plan de iteraciones. ....	27
Tabla 7. Plan de entrega de las iteraciones. ....	28
Tabla 8. Tarjeta CRC de la clase Adaline. ....	29
Tabla 9. Clases a sustituir. ....	32
Tabla 10. Historia de usuario HU2. ....	46
Tabla 11. Historia de usuario HU3. ....	46
Tabla 12. Historia de usuario HU4. ....	46
Tabla 13. Historia de usuario HU5. ....	47
Tabla 14. Historia de usuario HU6. ....	47
Tabla 15. Historia de usuario HU7. ....	47
Tabla 16. Historia de usuario HU8. ....	48
Tabla 17. Historia de usuario HU 9. ....	48
Tabla 18. Tarjeta CRC de la clase SimplePerceptron. ....	48
Tabla 19. Tarjeta CRC de la clase ID3. ....	49
Tabla 20. Tarjeta CRC de la clase ClassifierPanel. ....	49
Tabla 21. Tarjeta CRC de la clase PreprocessPanel. ....	50
Tabla 22. Tarjeta CRC de la clase ClusterPanel. ....	50
Tabla 23. Tarjeta CRC de la clase SimpleKMeans. ....	51
Tabla 24. Tarjeta CRC de la clase TreeVizualize. ....	51
Tabla 25. Tarjeta CRC de la clase VizualizePanel. ....	51
Tabla 26. Tarjeta CRC de la clase Attribute. ....	52
Tabla 27. Caso de Prueba para Gráfico de Pastel. ....	53
Tabla 28. Caso de prueba del AD ID3. ....	54
Tabla 29. Caso de Prueba para Perceptrón simple. ....	55
Tabla 30. Caso de Prueba para Adaline. ....	56
Tabla 31. Caso de Prueba para la red de Perceptrón simple. ....	57
Tabla 32. Caso de Prueba para la red de Adaline. ....	58
Tabla 33. Caso de Prueba para el algoritmo ID3. ....	59
Tabla 34. Caso de Prueba para el algoritmo K-Means. ....	60
Tabla 35. Caso de Prueba para el algoritmo K-NN. ....	61

## **INTRODUCCIÓN**

Con el acelerado avance de la sociedad moderna y el desarrollo empresarial alcanzado, surge la necesidad de almacenar los datos, interactuar con estos y clasificarlos en pequeños grupos que describan sus características principales, basándose en la similitud o diferencia entre ellos. Un gran porcentaje de la información generada representan “hechos” que diariamente se registran, tales como: transacciones financieras, operaciones de compra y venta, préstamos o devoluciones y movimientos de almacén.

Transformar los datos en información útil para la toma de decisiones es una tarea compleja, la transformación de los datos presenta una gran utilidad para el análisis y el apoyo en la toma de decisiones. Con el propósito de extraer la mayor cantidad de información provechosa para las empresas o instituciones, nace una nueva generación de técnicas computacionales y sistemas informáticos, los cuales constituyen el centro del emergente y dinámico campo de investigación denominado: minería de datos (MD). La MD es una de las fases del proceso de extracción de conocimientos a partir de los datos. Esta incluye diferentes técnicas de aprendizaje automático, la estadística, las bases de datos, los sistemas de toma de decisiones, la Inteligencia Artificial (IA) y otras áreas de la informática y de la gestión de información. Esto permite que las entidades o empresas minimicen los riesgos a la hora de decidirse estratégicamente, lo que trae consigo un ahorro considerable de tiempo, recursos humanos y financieros a la organización.

La MD la definió Mariela Maneiro como *“el descubrimiento eficiente de información valiosa, no-obvia de una gran colección de datos, cuyo objetivo es ayudar a buscar situaciones interesantes con los criterios correctos”* (Maneiro, 2008). Otros autores hacen referencia a la MD como el proceso de detección, análisis y extracción de conocimiento vital para la toma de decisiones (Cabena, et al., 1998).

La Sociedad Cubana de Reconocimiento de Patrones y Minería de Datos ha fomentado el intercambio de información entre expertos nacionales y extranjeros, permitiendo que actualmente se desarrollen proyectos en diferentes centros investigativos de todo el país y se produzcan aplicaciones de alta competitividad.

Algunas herramientas que se pueden utilizar para hacer MD son Orange, RapidMiner, WEKA, JHepWork y Knime. Es válido destacar el software WEKA por la cantidad de algoritmos para el análisis de datos que tiene implementado y por soportar gran cantidad de tareas, tales como preprocesamiento de datos, clasificación, agrupación, regresión, visualización y características de selección (JMACOE, 2011).

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Tema 2 Aprendizaje Automático de la asignatura de Inteligencia Artificial 2 (IA2) se estudian las diferentes fases del proceso de MD y se utiliza el software WEKA para realizar los pasos desde la carga de los datos, su transformación, la aplicación de algoritmos de aprendizaje automático y la comprensión de los modelos obtenidos. Este software contiene un gran número de algoritmos y tiene una estructura bien organizada, lo que facilita la navegación y la reutilización del código. Sin embargo, no tiene implementado los algoritmos de Redes Neuronales Perceptrón simple, ni Adaline. La herramienta (WEKA) brinda una visualización gráfica del funcionamiento de estos algoritmos, pero no muestra la ejecución de los pasos, ni cómo funciona internamente el Árbol de Decisión (AD), específicamente el ID3<sup>1</sup> (por sus siglas en inglés), K-Medias (K-Means), K-Vecino Más Cercano (K-NN<sup>2</sup>, por sus siglas en inglés) y Redes Neuronales. Ante la problemática planteada se define el siguiente **problema de la investigación**: ¿Cómo incorporar al software WEKA nuevas opciones de visualización de los algoritmos de análisis de datos estudiados en la asignatura Inteligencia Artificial 2? Teniendo como **objeto de estudio**: Opciones de visualización de los algoritmos de análisis de datos. Centrándose en el **campo de acción**: Opciones de visualización de los algoritmos de análisis de datos estudiados en Inteligencia Artificial 2 en el software de minería de datos WEKA.

Para resolver el problema planteado se propone como **objetivo general**: Desarrollar nuevas opciones de visualización de los algoritmos de análisis de datos Árboles de Decisión ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales presentes en el software de minería de datos WEKA.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de la investigación**:

- 1- Investigación y estudio de conceptos, herramientas y técnicas utilizadas para la MD.
- 2- Análisis de esquemas de visualización utilizados en herramientas de MD haciendo énfasis en los algoritmos de clasificación y agrupamiento para el software WEKA.
- 3- Desarrollo de nuevas opciones de visualización en el software WEKA para los algoritmos Árboles de Decisión ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales.

---

<sup>1</sup> Induction Decision Trees.

<sup>2</sup>K-Nearest Neighbor.

- 4- Validación de las funcionalidades definidas para las opciones de visualización a los algoritmos de análisis de datos estudiados en IA Árboles de Decisión ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales.

Durante la investigación se utilizaron los métodos de investigación teóricos y empíricos.

Los métodos teóricos que se utilizaron son:

- ❖ **Análisis Histórico-Lógico:** Durante la investigación se pone de manifiesto, en la realización de estudios de las causas que originaron el problema que presenta el software WEKA en la visualización de los algoritmos que se estudian en la asignatura de Inteligencia Artificial 2, así como para el análisis de las técnicas y algoritmos existentes en la actualidad para la MD.
- ❖ **Analítico-Sintético:** A través de este método se extraen las características, rasgos y elementos más importantes de las diferentes técnicas y herramientas aplicadas a la MD para la obtención de patrones y plantear la solución propuesta.

En el caso de los empíricos fueron utilizados:

- ❖ **Entrevista:** Este método se utiliza con el objetivo de obtener información valiosa que permita alcanzar el conocimiento necesario para dar solución al problema planteado.
- ❖ **Análisis Documental:** Se aplicó para el estudio de las distintas fuentes de información y con ello determinar los elementos necesarios y suficientes para desarrollar las opciones de visualización de los algoritmos de análisis de datos Árboles de Decisión ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales presentes en el software de minería de datos WEKA.

El trabajo de diploma se ha estructurado en tres capítulos:

**Capítulo 1: Fundamentación Teórica:** En este capítulo se abordan definiciones y conceptos importantes de la MD y se describen características, ventajas y desventajas de las técnicas y herramientas que se utilizan para dar solución al problema expuesto en la presente investigación.

**Capítulo 2: Análisis y diseño de la solución:** En este capítulo se detalla la propuesta de solución, se describe la arquitectura y la relación de los paquetes WEKA a través de un diagrama de paquete. Además se definen los requisitos funcionales y no funcionales.

**Capítulo 3: Descripción de los resultados:** En este capítulo se abordan los aspectos fundamentales del proceso de desarrollo y pruebas. Se define el estándar de codificación utilizado, además de la fase de implementación de la investigación y en la fase de pruebas se ejecutarán pruebas unitarias y de aceptación como forma de validación funcional de la propuesta de solución.



## **CAPÍTULO 1: FUNDAMENTOS TEÓRICOS**

En este capítulo se exponen los principales conceptos que sirven de base para adentrarse en el tema de la MD e Inteligencia Artificial, se caracterizan los algoritmos de MD estudiados en clases tales como Árboles de Decisión ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales. Se describen e identifican las herramientas y tecnologías empleadas para la solución, al igual que la metodología de desarrollo de software a utilizar para guiar el proceso de desarrollo de las opciones de visualización.

### **1.1 Minería de datos**

La MD es un campo de la ciencia computacional, la cual intenta descubrir patrones en grandes volúmenes de datos (Cabena, et al., 1998). El objetivo general de esta disciplina consiste en la extracción y transformación de la información en una estructura comprensible para su uso posterior. Con el paso del tiempo la investigación de la MD se ha hecho más extensa y autores como Mario Galvis y Fabricio Martínez en su libro titulado “Confrontación de dos técnicas de Minería de Datos aplicadas a un dominio específico.” se han referido a la MD como:

*“el conjunto de técnicas y herramientas aplicadas al proceso trivial de extraer y presentar el conocimiento implícito, previamente desconocido, potencialmente útil y humanamente comprensible, a partir de grandes conjuntos de datos, con el objeto de predecir de forma autorizada tendencias y comportamientos y/o descubrir de forma automatizada modelos previamente desconocidos”* (Galvis, y otros, 2004).

*“Una tecnología para el desarrollo y el descubrimiento de la información muchas veces oculta en los mismos datos. Información que muchas veces no se tiene en cuenta y que en determinados casos puede ser valiosa y crítica para un mejor conocimiento del negocio y aportar mayor base a la toma de decisiones en cualquier tipo de escenario”* (Galvis, y otros, 2004).

A partir de los conceptos analizados, los autores del presente trabajo definen como concepto de MD como: una tecnología capaz de descubrir y extraer información oculta que resulta valiosa para la toma de decisiones en distintos ámbitos.

#### **1.1.1 Áreas en las que se utiliza la MD**

La MD se puede aplicar en diversos sectores de la sociedad (Molina López, y otros, 2006), tal y como se define a continuación:

- Comercio y banca: segmentación de clientes, previsión de ventas, análisis de riesgo.
- Medicina y farmacia: diagnóstico de enfermedades y efectividad de los tratamientos.

- Seguridad y detección de fraude: reconocimiento facial, identificaciones biométricas, accesos a redes no permitidas.
- Recuperación de información no numérica: minería de texto, minería web, búsqueda e identificación de imagen, video, voz y texto de bases de datos multimedia.
- Astronomía: identificación de nuevas estrellas y galaxias.
- Geología, minería, agricultura y pesca: identificación de áreas de uso para distintos cultivos o de pesca o de explotación minera en bases de datos de imágenes de satélites.
- Ciencias ambientales: identificación de modelos de funcionamiento de ecosistemas naturales o artificiales para mejorar su observación, gestión y control.
- Ciencias sociales: estudio de los flujos de la opinión pública.

### **1.1.2 Fases de la MD**

La MD consta de 5 fases fundamentales, las cuales se encargan de definir las características del proceso (Centeno, y otros, 2011).

#### **1- Comprensión del negocio y del problema:**

Esta fase es una de las más delicadas, debido a que de ella depende la efectividad del proceso, pues los datos contenidos en la fuente tienen un formato que frecuentemente no es el adecuado, sobre el cual es muy difícil la aplicación de algún algoritmo de minería, por tanto los datos iniciales requieren ser limpiados y transformados para eliminar el ruido y los errores humanos. Este proceso se debe realizar teniendo en cuenta la integridad de los datos y del sistema de manera tal que no afecte su funcionamiento ni la lógica del negocio.

#### **2- Filtrado de datos:**

El proceso de filtrado se realiza porque es muy difícil la aplicación de los algoritmos de MD sobre los datos en bruto. Dependiendo de las características de la fuente y los algoritmos a utilizar, se eliminan valores no válidos o innecesarios, también se reduce el número de valores posibles mediante métodos de agrupamiento y redondeo, ajustándose así a las necesidades del algoritmo.

#### **3- Selección de variables:**

Luego del preprocesamiento y reducción de los datos, por lo general queda una gran cantidad de información para analizar. El principal objetivo de este nuevo proceso es reducir la dimensionalidad del espacio de las variables seleccionando sobre todas, las relevantes, con estas se construye un conjunto más pequeño, aplicando una transformación lineal o no lineal al conjunto original.

Para seleccionar este nuevo subconjunto se pueden utilizar dos métodos, el primero: los que priorizan la selección de los mejores atributos para el problema y segundo: los que mediante algoritmos de distancia, heurística o test de sensibilidad, encuentran las variables independientes.

#### **4- Extracción del conocimiento:**

Existen varios métodos de extracción de la información procedente de los datos que permite una mayor comprensión de estos y su reorganización en bases de datos. Esto posibilita caracterizar los datos de una manera simple y descubrir relaciones entre los datos espaciales y no espaciales. Estos métodos se separan en 5 grupos:

- ❖ Los basados en generalización, los cuales requieren la implementación de jerarquías de conceptos.
- ❖ Los de reconocimiento de patrones porque pueden ser usados para reconocimiento y categorización de fotografías, imágenes y textos.
- ❖ Los que usan agrupamiento, estos crean agrupamientos o asociaciones de datos cuando existen similitud entre ellos.
- ❖ Los que exploran asociaciones especiales, permiten descubrir reglas de asociaciones, las cuales asocian uno o más objetos con otro u otros objetos especiales.
- ❖ Los que utilizan aproximación y agregación que se encargan de descubrir conocimiento en base a las características representativas del conjunto de datos.

#### **5- Interpretación y evaluación:**

Después de aplicar todas las fases anteriores para obtener un modelo, es necesario validar el mismo, comprobando la autenticidad de las conclusiones, si estas son suficientemente satisfactorias, entonces la efectividad también sería satisfactoria. En caso de que se generen disímiles modelos, es necesario realizar una comparación entre cada uno de ellos, para determinar cuáles de estos se ajustan más al problema planteado, en caso de no alcanzar el resultado esperado, entonces es necesario modificar algunos de los pasos anteriores y así generar nuevos y mejores resultados.

### **1.2 Inteligencia Artificial**

La IA fue introducida a la comunidad científica en 1950 por el inglés Alan Turing en su artículo "Maquinaria Computacional e Inteligencia". La pregunta básica que Turing trató de responder afirmativamente en su artículo era: *¿pueden las máquinas pensar?* Una de las contribuciones más importantes de este inglés fue el diseño de la primera

computadora capaz de jugar ajedrez. Desde sus orígenes, la IA se relacionó con juegos de mesa debido a que estos construyen modelos de situaciones reales en las que hay que calcular, solucionar problemas, tomar decisiones, corregir los errores y recordar.

La IA es la parte de la ciencia que se ocupa del diseño de sistemas de computación inteligentes, los cuales reflejan o imitan características asociadas a la inteligencia humana haciendo referencia a la comprensión del lenguaje, el aprendizaje, el razonamiento y la resolución de problemas (Sharkey, 2012).

### **Áreas de la IA**

La IA ha ido creciendo y poco a poco se fue implementando en diversas áreas, algunas de estas son las siguientes (Gómez, y otros, 2013):

- **Robótica:** está muy vinculada con el uso de dispositivos inteligentes que mejoran las actividades humanas.
- **Medicina:** en esta ciencia se han desarrollado máquinas que interpretan imágenes médicas, controlan las unidades de cuidados intensivos, monitorean a los pacientes y realizan diagnósticos. Actualmente se han desarrollado máquinas que detectan las enfermedades posibles a mediano plazo de un paciente y así prevenir muchas de ellas.
- **Educación:** en el campo de la educación se ha hecho necesario incluir la IA debido al gran trabajo que generan extensas cantidades de estudiantes, porque se han implantado sistemas de gestiones de estudiantes para reducir el estrés que genera la misma, como también otro tipo de sistemas que puedan percibir las deficiencias de un estudiante y ayudar en su desenvolvimiento.
- **Entretenimiento:** la IA en los videojuegos es cada vez más indispensable tanto en consolas como en ordenadores aunque los usuarios no la distinguen debido a que está muy implícita, la IA se evidencia en los famosos avatares y textos.

### **1.3 Algoritmos de MD**

Los algoritmos de MD provienen de la IA, los cuales se aplican sobre un conjunto de datos para obtener resultados. Algunos de ellos se describen a continuación.

#### **1.3.1 Árbol de Decisión (AD)**

Los Árboles de Decisión son algoritmos que de una forma gráfica y analítica representan todos los eventos (sucesos) que pueden surgir a partir de una decisión asumida en cierto momento (Peng, et al., 2012). Ayudan a tomar las decisiones “más acertadas”, desde un punto de vista probabilístico, a partir de un conjunto de posibles decisiones (Estadística). Uno de los algoritmos de Árbol de Decisión que tiene implementado el software WEKA es el ID3. El AD ID3 es un modelo de clasificación que significa

"Inducción mediante árboles de decisión", capaz de tomar decisiones con gran precisión. Es un algoritmo muy rápido, tiene como objetivo construir un AD que explica cada instancia de la secuencia de entrada de la manera más compacta posible a partir de una tabla de inducción.

En cada momento elige el mejor atributo dependiendo de una determinada ganancia de información, determina las variables que portan información relevante para la solución del problema y establece la secuencia dentro del AD. Como características principales destacan: que es recursivo, no realiza "backtracking", utiliza la entropía (es la medida de la incertidumbre que hay en un sistema, es decir, ante una determinada situación, la probabilidad de que ocurra cada uno de los posibles resultados) y aplica la estrategia "divide y vencerás" para la clasificación de los objetos y asociarlos a una clase (López Takeyas, 2013).

### **1.3.2 K-Means**

El K-Means es un algoritmo de agrupamiento, que tiene como objetivo la partición de un conjunto de  $n$  observaciones en  $k$  grupos, en el que cada observación pertenece al grupo más cercano a la media. El proceso de agrupamiento K-Means es simple, inicialmente se determina el número de grupos  $k$  y se calcula el centroide o centro de esos grupos. Para determinar los centroides hay dos alternativas prácticas: la primera es tomar de forma aleatoria  $k$  objetos como centroides iniciales y la segunda es tomar los primeros  $k$  objetos en secuencia. Luego el algoritmo ejecuta los siguientes tres pasos hasta que alcance el criterio de convergencia, es decir, que los objetos no se muevan de grupo. Como primer paso se determina el o los centroides iniciales de acuerdo al número de cluster<sup>3</sup> esperados. En segundo lugar se determina la distancia de cada objeto con relación a los centroides y por último se agrupan los objetos con base en la distancia mínima (Pascual, y otros, 2010).

### **1.3.3 K-Vecino más cercano**

K-Vecino más cercano es un método de vecindad basado en caso o instancias, pertenece al grupo de métodos para tareas de clasificación de datos. Su funcionamiento se basa en que supone que los vecinos más cercanos proponen la mejor clasificación y esto se hace utilizando todos los atributos. El problema de dicha suposición es que es muy probable que se tengan muchos atributos irrelevantes que dominen sobre la clasificación: dos atributos relevantes perderían peso entre veinte irrelevantes. Parte de la idea de que una nueva muestra será clasificada a la clase a la cual pertenezca la mayor cantidad de vecinos más cercanos.

---

<sup>3</sup> Cluster: grupo

En problemas prácticos donde se aplica este algoritmo de clasificación se acostumbra tomar un número  $k$  impar de vecinos para evitar posibles empates, aunque esta forma es cierta en problemas que poseen dos clases nada más. Los empates se pueden resolver mediante la clasificación aleatoria de la muestra entre las clases empatadas o la clase donde la distancia media de sus vecinos sea inferior (García Cambrónero, y otros, 2006).

#### **1.3.4 Redes neuronales**

Las Redes Neuronales son un conjunto de neuronas conectadas entre sí sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para "aprender" algo que se queda fijo en el tejido. Las Redes Neuronales permiten buscar la combinación de parámetros que mejor se ajusta a un determinado problema, son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. En el lenguaje propio, encontrar la combinación que mejor se ajusta es "entrenar" la red neuronal. Una red ya entrenada se puede usar luego para hacer predicciones o clasificaciones, es decir, para "aplicar" la combinación (Matich, 2011).

- **Perceptrón simple:** El Perceptrón está constituido por un conjunto de sensores de entrada que reciben los patrones a reconocer o clasificar y una neurona de salida que se ocupa de clasificar a los patrones de entrada en dos clases, según la salida de la misma, sea activada (1) o desactivada (0). El Perceptrón utiliza la salida de la función umbral (binaria) para el aprendizaje. Sólo se tiene en cuenta si se ha equivocado o no. Una desventaja que posee el Perceptrón simple es que solo puede resolver problemas que son linealmente separables (Tomás Mariano, 2011).
- **Adaline:** La red Adaline tiene una estructura similar al Perceptrón, la diferencia es la manera de utilizar la salida en la regla de aprendizaje. En Adaline se utiliza directamente la salida de la red (real) teniendo en cuenta la diferencia entre el valor real esperado ( $d$ ) y la salida producida por la red ( $y$ ). Será imposible conseguir una salida exacta porque ( $y$ ) es una función lineal, pero se minimizará el error cometido para todos los patrones de entrenamiento. La regla de aprendizaje será la regla Delta. Esta regla busca el conjunto de pesos que minimiza la función de error mediante un proceso iterativo donde se van presentando los patrones uno a uno y se van modificando los parámetros de la red (Chavéz Solano, 2010).

## **1.4 Herramientas de MD**

Las herramientas de MD se utilizan para explorar los datos que se encuentran en las bases de datos o almacenes de datos, que algunas veces contienen información almacenada durante varios años. Las herramientas de MD tienen un papel importante, porque ayudan a extraer información que resulta valiosa. La información obtenida se combina, se analiza y procesa rápidamente. A continuación se describe la herramienta que se utiliza en la asignatura IA2.

### **1.4.1 WEKA**

Se trata de una herramienta totalmente desarrollada en el lenguaje de programación Java, que contiene una interfaz gráfica para la aplicación de los algoritmos directamente al conjunto de datos de origen, así como la posibilidad de aplicarlos desde un programa Java independiente (Morate, 2007).

Contiene herramientas para el preprocesamiento de los datos, clasificación, regresión, agrupamiento, reglas de asociación y visualización. Asimismo, facilita la creación de nuevos esquemas de aprendizaje mediante la utilización de los algoritmos ya implementados.

Es un software de código abierto y funciona bajo el esquema de GNU<sup>4</sup>. Fue creado en el Departamento de Ciencias de la Computación de la Universidad de Waikato, en Nueva Zelanda (Hernández Orallo, y otros, 2006). El software WEKA presenta una serie de características entre las que se encuentran:

- ❖ Posee un grupo de herramientas que permiten el análisis de los datos.
- ❖ Tiene implementados algoritmos de clasificación, agrupamiento y reglas de asociación.
- ❖ Al cargar los datos, WEKA realiza un análisis de los mismos y obtiene las mejores variables para obtener el modelo de conocimiento.
- ❖ La licencia de WEKA es GPL<sup>5</sup>, lo que significa que este programa es de libre distribución y difusión.

#### ➤ **Ventajas:**

- ❖ Se trata de un software específicamente diseñado y utilizado para investigación y fines educativos. Por esta razón, los elementos que brinda como salida, no están orientados exclusivamente hacia la obtención de herramientas para el dominio de aplicación, sino también hacia la obtención

---

<sup>4</sup> GNU: es un sistema operativo de tipo Unix desarrollado por y para el Proyecto GNU. GNU es un acrónimo recursivo de "GNU's Not Unix!" (en español: GNU no es Unix).

<sup>5</sup> GPL: Licencia Pública General- General Public License.

de información acerca del proceso de minería y de la calidad de los resultados obtenidos.

- ❖ Es una herramienta bajo el esquema de licenciamiento público, su uso es totalmente gratis, lo cual facilita su aprovechamiento

#### 1.4.2 Desventajas de la visualización de los algoritmos en el software de MD WEKA.

Las principales deficiencias que presenta el software WEKA en la visualización de los algoritmos AD ID3, K-Medias, K-Vecinos más cercanos y Redes Neuronales Adaline y Perceptrón simple son:

- **Preprocesamiento:** En esta etapa la visualización de los datos solo muestra un histograma para la comparación de los atributos.
- **AD ID3:** Este algoritmo no muestra el árbol de decisión que devuelve la ejecución del mismo, tampoco muestra la forma en las que se seleccionan los atributos para ir creando el árbol por niveles.
- **Redes Neuronales:** El software no cuenta con los algoritmos de Redes Neuronales Adaline y Perceptrón simple, por tal motivo no muestra la red neuronal que se obtiene de la ejecución de los mismos ni la tabla donde se muestra la forma en varían los pesos y el *bias* a medida que evalúa cada instancia.
- **K-Vecino más cercano:** Este algoritmo muestra la cantidad de instancias que se clasifican correctamente y las que no, dicho algoritmo no refleja los vecinos más cercanos para cada una de las instancias, ni la distancia a cada uno de los vecinos, ni cómo se clasifican la instancias después de aplicar el algoritmo al conjunto de datos.
- **K-Medias:** Este algoritmo solo muestra la cantidad de *cluster* que presenta y la cantidad de instancias que se agrupan en cada uno de ellos, pero no refleja cuales son estas instancias ni los valores de las mismas.

#### 1.5 Lenguaje de programación

Los lenguajes de programación están diseñados para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras, además de usarse para crear programas que controlen el comportamiento físico y lógico de una computadora. Los lenguajes de programación están formados por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.



### **1.5.1 Java**

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases. Este lenguaje 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, por lo que el código que se ejecuta en una plataforma no tiene que ser recompilado para correr en otra. Java presenta un grupo de características que se enuncian a continuación:

- Los programas Java no son ejecutables, no se compilan como los programas en C o C++. En su lugar son interpretados por una aplicación conocida como la máquina virtual de Java (JVM). Gracias a ello no tienen por qué incluir todo el código y librerías propias de cada sistema.
- La finalidad de Java es crear aplicaciones de todo tipo (móvil, de escritorio, web) aunque está condicionado para crear sobre todo aplicaciones en red.
- Java elimina las instrucciones dependientes de la máquina y los punteros que generan terribles errores en C y la posibilidad de generar programas para atacar sistemas (Sánchez, 2004).
- Algunas de las aplicaciones que se han desarrollado en el lenguaje Java son WEKA, JUnit, Knime y JHepWork.

### **1.6 Entorno de Desarrollo Integrado (IDE)**

Un IDE es una aplicación compuesta por un conjunto de herramientas útiles para un programador. Un entorno IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador, y un constructor de interfaz gráfica GUI<sup>6</sup> (Alegsa, 2010). A continuación se hace un análisis de los principales IDE para el desarrollo en el lenguaje de programación Java y seleccionar el más adecuado para la solución del problema.

#### **1.6.1 Eclipse**

Eclipse es un IDE de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. En este se encontrarán algunas de las herramientas y funciones necesarias para trabajar, recogidas además en una interfaz que lo hace fácil de usar (López Martínez, y otros, 2011).

- **Ventajas:**

---

<sup>6</sup> Graphic User Interface o Interfaz Gráfica de Usuario.

- ❖ Eclipse emplea módulos (*plugins*) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.
- ❖ Eclipse permite utilizar lenguajes de procesamiento de texto, aplicaciones de red, Sistemas de Gestión de Bases de Datos.
- ❖ Brinda soporte para Sistemas de Control de Versiones e incluye extensiones para realizar pruebas de unidad.
- **Desventajas:**
  - ❖ Algunas aplicaciones pueden llegar a ser algo complicadas de instalar.
  - ❖ A nivel de hardware exige muchos recursos.
  - ❖ Poca estabilidad y flexibilidad en el campo de multimedia y juegos.
  - ❖ Menor compatibilidad con el hardware.

### **1.6.2 NetBeans**

NetBeans es un proyecto de código abierto de gran éxito con una amplia base de usuarios, una comunidad en constante crecimiento. *Sun Microsystems* fundó el proyecto de código abierto en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. NetBeans constituye una solución muy completa para programar en Java, aunque soporta otros lenguajes como C/C++, JavaScript, Ruby, Groovy, Python y PHP. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación y consta de una gran cantidad de módulos (De Castro Riveras, 2009).

- **Ventajas:**
  - ❖ NetBeans es multilenguaje, utilizado tanto por programadores con poca experiencia como por expertos.
  - ❖ Fue creado por la propia compañía fundadora de Java, *Sun Microsystems*.
  - ❖ Permite trabajar sobre el mismo código a más de un programador.
  - ❖ Cuenta con excelente soporte para aplicaciones web.

### **1.6.3 Fundamentación del IDE seleccionado**

El IDE seleccionado es NetBeans. Una de sus características es la simplicidad y la experiencia que tienen los autores de este trabajo con este IDE. Además de ser un producto libre y gratuito sin restricciones de uso, hecho principalmente para el lenguaje de programación Java. NetBeans es un lenguaje, utilizado tanto por programadores con poca experiencia como por expertos. Fue creado por la propia compañía que creó el lenguaje de programación Java, *Sun Microsystems*.

## **1.7 Herramienta de modelado**

### **1.7.1 Visual Paradigm**

Visual Paradigm es una herramienta para el desarrollo de aplicaciones utilizando modelado UML<sup>7</sup>, ideal para ingenieros de software, analistas y arquitectos de sistemas que están interesados en construcción a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. También ofrece navegación intuitiva entre la escritura del código y su visualización; potente generador de informes en formato PDF/HTML<sup>8</sup>. Visual Paradigm presenta un ambiente visualmente superior de modelado y sincronización de código fuente en tiempo real (Gutiérrez, 2009). Entre las principales características de esta herramienta se encuentran:

- ❖ Permite la generación de código e ingeniería tanto directa como inversa.
  - ❖ Utiliza el lenguaje de modelado UML para visualizar y diseñar los elementos de software.
  - ❖ Los desarrolladores pueden diseñar la documentación del sistema con una plantilla de diseño.
  - ❖ El analista de sistemas pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis.
  - ❖ Se integra con las siguientes herramientas Java:
    - Eclipse/IBM WebSphere.
    - Jbuilder.
    - Netbeans IDE.
    - Oracleeblogic.
    - Jdeveloper.
    - BEA W.
- **Ventajas:**
- ❖ Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
  - ❖ Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
  - ❖ Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.

---

<sup>7</sup> UML: Lenguaje Unificado de Modelado (por sus siglas en inglés, Unified Modeling Language.)

<sup>8</sup> PDF/HTML: Portable Document Format/HyperText Markup Language.

- ❖ Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
  - ❖ Generación de código e ingeniería inversa: genera el código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
  - ❖ Generación de documentación: permite documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- **Desventajas:**
- ❖ Las imágenes y reportes generados, no son de muy buena calidad. Instalación costosa.
  - ❖ Los modelos a veces no pueden ser reabiertos.
  - ❖ No hay llamadas reflexivas en los diagramas de secuencia.
  - ❖ Se debe seleccionar una clase para crear un diagrama de secuencia.

### **1.7.2 Enterprise Architect**

Es una herramienta de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Enterprise Architect es una herramienta multiusuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad (Spaerxs system, 2013). Las principales características de Enterprise Architect son:

- ❖ Crea elementos del modelo UML para un amplio alcance de objetivos y los documenta.
  - ❖ Ubica esos elementos en diagramas y paquetes.
  - ❖ Crea conectores entre elementos.
  - ❖ Genera código para el software que está construyendo.
  - ❖ Realiza ingeniería reversa del código existente en varios lenguajes.
  - ❖ Importación/Exportación XML 2.1.
  - ❖ Nuevo motor de reporte HTML.
- **Ventajas:**
- ❖ Es un software de alta capacidad, flexible y de mucha calidad.
  - ❖ Cuenta con gran velocidad, estabilidad y buen rendimiento a la hora de construir modelos de sistemas de software rigurosos donde es posible mantener la trazabilidad de manera consistente. Enterprise Architect soporta este proceso en un ambiente fácil de usar, rápido y flexible.

- ❖ Trazabilidad de extremo a extremo, provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue.
- **Desventajas:**
  - ❖ No tiene gran calidad de interfaz.
  - ❖ El soporte de ayuda es de manejo intuitivo, o sea, no es específico; lo cual dificulta su utilización.

### **1.7.3 Rational Rose Enterprise Edition**

Es una herramienta desarrollada por *Rational Corporation* basada en UML, permite crear los diagramas que se generan durante el proceso de ingeniería de software. También posibilita el modelado en ese lenguaje para el diseño de las bases de datos, que integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos. Brinda funciones de análisis de calidad de código y posee un complemento de modelado web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones web (González Blanco, y otros, 2010).

Entre las características principales de Rational se pueden destacar:

- ❖ Permite desarrollo multiusuario.
- ❖ Genera documentación del sistema.
- ❖ Disponible en múltiples plataformas.
- **Ventajas:**
  - ❖ Permite una creación más rápida de software de calidad.
  - ❖ Proporciona prestaciones de modelado visual para desarrollar muchos tipos de aplicaciones de software.
  - ❖ Contiene herramientas web y XML para el modelado de aplicaciones web.
  - ❖ Unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes.
- **Desventajas:**
  - ❖ Rational Rose no es una herramienta gratuita.

### **1.7.4 Fundamentación de la herramienta de modelado seleccionada**

La herramienta de modelado seleccionada es Visual Paradigm por ser muy potente y soportar el ciclo de vida completo de desarrollo del software. Se puede modelar todos los tipos de diagramas, generar código y documentaciones de estos. También permite ejecutar un modelo UML en cualquier plataforma utilizando el mismo programa sin ser afectado. Además destacar que guarda todo el trabajo en un mismo fichero, trabaja muy

bien en ordenadores poco potentes y es una herramienta sin coste alguno, completamente gratis.

### **1.8 Metodología de desarrollo de software**

Las metodologías de desarrollo se utilizan para estructurar, controlar y planificar el proceso en sistemas de información. En la actualidad existen varias propuestas, entre las cuales se encuentran las tradicionales que se basan específicamente en el control de proyectos de gran tamaño. Sin embargo, a medida que pasan los años se ha demostrado que las metodologías tradicionales no ofrecen una correcta solución a proyectos donde los requisitos no se conocen con exactitud. Debido a esta dificultad presentada surgen las metodologías ágiles las cuales intentan adaptarse a la realidad del software (Hernández Muñoz, 2006)

#### **1.8.1 Programación Extrema**

XP (*eXtreme Programming*) es una metodología ágil creada para el desarrollo de software. Se usa para satisfacer las necesidades del cliente. Está diseñada fundamentalmente para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, de esta forma la comunicación entre ellos es mucho más factible y es muy importante que la realicen entre los programadores, los jefes de proyectos y los clientes (Pressman, 2007). Entre las características más relevantes de XP están:

- ❖ La comunicación que se tiene entre los programadores y los clientes para así satisfacer sus requisitos y responder rápidamente a los cambios que los mismos necesiten.
- ❖ La simplicidad que se encarga de la codificación, y los diseños simples y claros.
- ❖ Por último la retroalimentación mediante la cual se ofrece al cliente la posibilidad de ir viendo el proyecto y así conseguir un sistema apto a sus necesidades.

#### **1.8.2 Proceso Unificado de Desarrollo**

RUP (por sus siglas en inglés) es un proceso de desarrollo de software que define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. Requiere un grupo grande de programadores para trabajar con esta metodología, es el proceso de desarrollo más general de los existentes actualmente. El proceso RUP estima tareas y horarios del plan midiendo la velocidad de iteraciones, donde estas se enfocan fundamentalmente sobre la arquitectura del software (Jacobson, y otros, 1999). La

puesta en práctica de características se retrasa hasta que se ha identificado y se ha probado una arquitectura firme. Posee tres características fundamentales:

- ❖ **Dirigido por casos de uso:** Los Casos de Uso (CU) reflejan lo que los usuarios futuros necesitan y desean, constituyen la guía fundamental establecida para las actividades a realizar durante el proceso de desarrollo del sistema.
- ❖ **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.
- ❖ **Iterativo e incremental:** RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones.

### **1.8.3 Fundamentación de la metodología seleccionada**

La metodología seleccionada es XP al implementar una forma de trabajo que se adapta fácilmente a las características del presente trabajo, presenta una tasa de errores mínima, su programación es más organizada que la de cualquier otra metodología y está diseñada fundamentalmente para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño. Además esta metodología promueve soluciones simples que se ajustan a las necesidades específicas lo que permite el ahorro de tiempo en la búsqueda de la solución. Todos los miembros del equipo comparten el mismo objetivo para un desarrollo óptimo evitando así el fracaso del proyecto.

### **1.9 Patrones de diseño**

Los patrones de diseño describen un problema que ocurre una y otra vez en el entorno, para describir después el núcleo de la solución a ese problema, de tal manera que dicho desarrollo pueda usarse varias veces de la misma forma. El uso de estos patrones permite ahorrar grandes cantidades de tiempo en la construcción de software, tener una estructura de código común a todos los proyectos que implementen una funcionalidad genérica y es más fácil de comprender, mantener y extender un software que aplique patrones de diseño (Gamma, y otros, 1995).

*“Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular”* (Gamma, et al., 1995).

### 1.9.1 Patrones de asignación de responsabilidades o GRASP<sup>9</sup>

Estos describen los principios fundamentales de la asignación de responsabilidades a las clases y objetos. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (Cortés, y otros, 2010). Los patrones GRASP se exponen a continuación:

- **Bajo acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras clases. La información que almacena una clase debe de ser coherente y debe estar relacionada con la clase. Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.
- **Alta cohesión:** La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Con el uso de este patrón se mejora la claridad y la facilidad con que se entiende el diseño, se simplifican el mantenimiento. A menudo se genera un bajo acoplamiento.
- **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.
- **Creador:** Se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:
  - ❖ B contiene a A.
  - ❖ B es una agregación (o composición) de A.
  - ❖ B almacena a A.
  - ❖ B tiene los datos de inicialización de A (datos que requiere su constructor).
  - ❖ B usa a A.
- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

---

<sup>9</sup> GRASP: General Responsibility Assignment Software Patterns



### **1.9.2 Patrones GoF<sup>10</sup>**

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación. Cuando se publica el libro "Design Patterns: Elements of Reusable Object Oriented Software" escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, la idea de patrones de diseño comenzó a tomar fuerza. Ellos recopilaron y documentaron 23 patrones de diseño, los que proporcionan elementos reusables en el diseño de sistemas de software, por tanto se pueden aplicar a diferentes problemas de diseño en distintas circunstancias (García Mondaray, 2012). (Ver **Anexo 1: Patrones GoF**).

❖ **Patrón Instancia única o Singleton:**

Este patrón garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella, permitiendo controlar la misma mediante dicho acceso. Además reduce el espacio de nombres y facilita refinar operaciones. Es utilizado cuando se requiere de un acceso estandarizado y conocido.

### **1.10 Conclusiones parciales**

En este capítulo se expusieron las características más importantes de la MD y de la IA, se detallaron además las principales fases de la MD. Se abordaron las características de algunas herramientas que utiliza dicho campo para el análisis y exploración de información y por último se definieron las herramientas y tecnologías a utilizar, como metodología de desarrollo de software se eligió XP, el software de MD WEKA en su versión 3.6.1, como IDE NetBeans en su versión 8.0, como lenguaje de programación Java y para el modelado la herramienta case Visual Paradigm en su versión 8.0.

---

<sup>10</sup> GoF: Gang of Four, Pandilla de los Cuatro.

## **CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN**

Este capítulo tiene como objetivo describir las actividades desarrolladas durante todo el proceso de análisis y diseño. Se detalla la propuesta de solución, así como la relación que tienen los paquetes del WEKA mediante un diagrama de paquetes y se describe su arquitectura. Los requisitos funcionales y no funcionales, conjuntamente con los artefactos que plantea la metodología escogida contribuyeron al desarrollo de las nuevas funcionalidades agregadas.

### **2.1 Propuesta de solución**

Utilizando la información recopilada en el capítulo precedente, se propone el desarrollo de nuevas funcionalidades que implementen algoritmos y mejoren la visualización que presenta el software de MD WEKA que se utiliza para impartir las clases de IA2. La solución informática debe ser capaz de centralizar los problemas y deficiencias que presenta este software para que brinde un mejor servicio. Además debe permitirles a los profesores y estudiantes comprender la MD, así como el preprocesamiento de los datos a través de los algoritmos estudiados en las clases.

También, mediante gráficas y esquemas de visualización mejorar la comprensión del proceso de MD para los algoritmos de AD, específicamente ID3, para los de Redes Neuronales, Perceptrón simple y Adaline, así como para el algoritmo K-Means y K-NN.

### **2.2 Diagrama de paquetes**

El diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre las mismas. Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada uno y minimizar el acoplamiento externo entre ellos. En este trabajo el diagrama muestra la relación entre los paquetes que presenta el software WEKA para mejorar su comprensión y entendimiento (Gutiérrez, 2009). (Ver **Anexo 2: Diagrama de paquetes**).

A continuación se describen los paquetes de este software:

- **Núcleo:** este paquete provee, estructuras y métodos necesarios para el correcto funcionamiento de los demás paquetes del WEKA.
- **Interfaz gráfica de usuario:** es una parte fundamental de la apariencia del sistema porque controla la mayoría de las vistas del software, así como todos los paneles y gráficas que permiten la interacción y comprensión de los algoritmos con que este programa interactúa.

- **Filtrado:** este paquete es el encargado de procesar los datos con que trabajan los distintos algoritmos que contiene el WEKA, un ejemplo son los dataset<sup>11</sup>, es necesario porque los algoritmos del WEKA solo trabajan con números y los datos pueden ser nominales por tanto hay que filtrarlos.
- **Agrupamiento:** este paquete contiene toda la información necesaria para que el software ejecute los algoritmos de agrupamiento.
- **Clasificación:** paquete que contiene las clases necesarias para la ejecución de los algoritmos de clasificación.
- **Asociación:** paquete que contiene las clases que implementan los métodos para realizar asociaciones.
- **Selección de atributos:** paquete que permite aplicar diversas técnicas para la reducción del número de atributos según el algoritmo a emplear.
- **Generador de dato:** paquete que contiene clases abstractas para generadores de datos y clases para representar tests.
- **Estimadores:** este paquete contiene estimadores de probabilidades condicionadas y de probabilidades simples de diferentes distribuciones (Poisson, Normal).
- **Experimento:** paquete que contiene clases que implementan la entidad experimento. Así mismo, están las que modelan los experimentos remotos.

### 2.3 Requisitos del sistema

Un requisito es una “condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado” (Laguna, 2012). También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación. Existen dos tipos de requisitos, los funcionales y los no funcionales.

#### 2.3.1 Requisitos funcionales

Los requisitos funcionales denotan una funcionalidad del sistema, son característica requerida que expresan una capacidad de acción del mismo o una determinada función; generalmente expresada en una declaración en forma verbal. Sirve de base para estimar el coste, el tiempo necesario para desarrollar el software y para planificar los contenidos técnicos de las iteraciones posteriores (Pressman, 2007). A partir de la descripción de los procesos de negocios, se identificaron los requisitos a cumplir por el sistema, los cuales se muestran a continuación:

---

<sup>11</sup> Dataset: conjunto de datos.

No.	Requisitos funcionales
RF1	Agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.
RF2	Visualizar el árbol de decisión del algoritmo ID3.
RF3	Agregar el algoritmo para Redes Neuronales del Perceptrón simple.
RF4	Agregar el algoritmo para Redes Neuronales Adaline.
RF5	Visualizar la topología para el algoritmo Adaline.
RF6	Visualizar la topología para el algoritmo Perceptrón simple.
RF7	Visualizar la ejecución del algoritmo ID3.
RF8	Visualizar la ejecución y procesamiento del algoritmo K-Means.
RF9	Visualizar la ejecución y procesamiento del algoritmo K-NN.

**Tabla 1. Requisitos funcionales.**

### 2.3.2 Requisitos no funcionales

Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

➤ **Usabilidad:**

RNFU 1: Las nuevas funcionalidades deben presentar un acceso fácil y rápido, para facilitar su uso por los usuarios.

➤ **Diseño e implementación:**

RNFDI 1: Se utilizará el software de MD WEKA en su versión 3.6.1 como base para la arquitectura de las funcionalidades que se le agregaron al mismo.

RNFDI 2: Se empleará como IDE NetBeans en su versión 8.0 para el desarrollo de las funcionalidades.

RNFDI 3: Se utilizará Visual Paradigm en su versión 8.0 como herramienta de modelado.

➤ **Funcionamiento:**

• **Software:**

RNFS 1: Para el correcto funcionamiento en la PC será necesario tener instalado el sistema operativo GNU/Linux en cualquiera de sus distribuciones o el sistema operativo Windows XP o una superior, así como la máquina virtual de Java en su versión 7.0.

• **Hardware:**

RNFH 1: La PC para la correcta ejecución de la aplicación deberá tener las siguientes características mínimas:

- Procesador Intel Pentium Celeron a 2.0 Ghz, equivalente o superior.

- 1GB<sup>12</sup> de memoria RAM<sup>13</sup>.
- Capacidad de 10 GB de HDD<sup>14</sup>.

➤ **Fiabilidad:**

RNFF 1: Ante cualquier operación que se realice sobre las funcionalidades que se le agregaron al WEKA, este debe responder en no más de 6 segundos.

### 2.3.3 Lista de reserva del producto

Ítem*	Descripción	Estimación	Estimado por
<b>Prioridad: Muy Alta</b>			
RF 3	Agregar el algoritmo para Redes Neuronales del Perceptrón simple.	2 semanas	Analista
RF 4	Agregar el algoritmo para Redes Neuronales del Adaline.	2 semanas	Analista
RF 6	Visualizar la topología para el algoritmo Perceptrón simple.	2 semana	Analista
<b>Prioridad: Alta</b>			
RF 2	Visualizar el árbol de decisión del algoritmo ID3.	1 semanas	Analista
RF 7	Visualizar la ejecución del algoritmo ID3.	1 semanas	Analista
RF 5	Visualizar la topología del algoritmo Adaline	1 semanas	Analista
<b>Prioridad: Media</b>			
RF 1	Agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.	0.5 semanas	Analista
RF 8	Visualizar la ejecución y procesamiento del algoritmo K-Means.	0.5 semanas	Analista
RF 9	Visualizar la ejecución y procesamiento del algoritmo K-NN.	1.5 semanas	Analista

**Tabla 2. Lista de reserva del producto RF.**

Ítem *	Descripción	Estimación	Estimado por
<b>Prioridad: Muy Alta</b>			
RNF: Diseño e implementación.	Se utilizará el software de MD WEKA en su versión 3.6.1 como base para la arquitectura de las funcionalidades que se le agregaron al mismo.		Analista

<sup>12</sup> GB: Gigabyte, es una unidad de almacenamiento.

<sup>13</sup> RAM: Memoria de Acceso Aleatorio: Random Access Memory.

<sup>14</sup> HDD: Unidad de Disco Duro: Hard Disk Drive.

RNF: Diseño e implementación.	Se utilizará como IDE NetBeans en su versión 8.0 para el desarrollo de las funcionalidades.		Analista
RNF: Diseño e implementación.	Se utilizará Visual Paradigm en su versión 8.0 como herramienta de modelado.		Analista
<b>Prioridad: Alta</b>			
RNF: Funcionamiento.	Para el correcto funcionamiento en la PC será necesario tener instalado el sistema operativo GNU Linux.		Analista
RNF: Funcionamiento.	La PC para la correcta ejecución de la aplicación deberá tener las siguientes características mínimas: <ul style="list-style-type: none"> <li>• Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior.</li> <li>• 2GB de memoria RAM.</li> <li>• Capacidad de 80 GB de disco duro.</li> </ul>		Analista
<b>Prioridad: Media</b>			
RNF: Fiabilidad	Ante cualquier operación que se realice sobre las funcionalidades que se le agregaron al WEKA, este debe responder en no más de 6 segundos.		Analista
RNF: Usabilidad	Las nuevas funcionalidades deben presentar un acceso fácil y rápido, para simplificar su uso por los usuarios.		Analista

**Tabla 3. Lista de reserva del producto RNF**

## 2.4 Historia de usuario

Las historias de usuarios (HU) son utilizadas como herramientas para dar a conocer los requerimientos del sistema al equipo de desarrollo. Son pequeños textos para describir una actividad que realizará el software. Se puede considerar que estas juegan un papel similar a los casos de uso en otras metodologías, pero en realidad son muy diferentes porque solo muestran la silueta de una tarea a realizarse.

Las HU también son utilizadas para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas. En una entrega se puede desarrollar una o más HU, esto depende solo del tiempo que demore la implementación de cada una de las mismas (Pressman, 2007). (Ver **Anexo 3: Historia de Usuario**).

Historia de Usuario	
<b>Número:</b> HU 1	<b>Nombre Historia de Usuario:</b> Agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.
<b>Usuario:</b> Hayron Pérez	<b>Iteración Asignada:</b> 1
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 0.50
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 0.50
<b>Descripción:</b> La presente historia de usuario tiene como objetivo permitir agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.	
<b>Observaciones:</b>	

**Tabla 4. Historia de Usuario HU1.**

## 2.5 Fase de planificación

En esta fase el cliente establece la prioridad de cada HU, y teniendo en cuenta esto los programadores realizan una estimación de esfuerzo necesario para cada una de ellas. Se realizan acuerdos sobre el material a entregar en la primera iteración y en correspondencia se genera un cronograma junto al cliente (Pressman, 2007).

### 2.5.1 Estimación de esfuerzo

La medida utilizada para la estimación del esfuerzo asociado a la implementación es el punto. Un punto equivale a una semana ideal de programación. Esta medida generalmente toma valores de 1 a 3.

No.	Historia de usuario	Punto de estimación
1.	Agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.	0.50
2.	Visualizar el árbol de decisión del algoritmo ID3.	1
3.	Agregar el algoritmo para Redes Neuronales del Perceptrón simple.	2
4.	Agregar el algoritmo para Redes Neuronales Adaline.	2
5.	Visualizar la topología para el algoritmo Adaline.	1
6.	Visualizar la topología para el algoritmo Perceptrón simple.	2
7.	Visualizar la ejecución del algoritmo ID3.	1
8.	Visualizar la ejecución y procesamiento del algoritmo K-Means.	0.50
9.	Visualizar la ejecución y procesamiento del algoritmo K-NN.	1.50

**Tabla 5. Plan de esfuerzo por HU.**

### 2.5.2 Plan de iteraciones

Para cada entrega fueron escogidas las HU, teniendo en cuenta el orden definido. Este plan define las historias de usuario que deben ser implementadas en cada iteración y las fechas de liberación. A continuación se muestran 3 iteraciones y el número de historias por cada una.

Iteración	Historias de usuario	Duración total (semanas)
1	Agregar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.	2.50
	Visualizar el árbol de decisión del algoritmo ID3.	
	Visualizar la ejecución del algoritmo ID3.	
2	Agregar el algoritmo para Redes Neuronales del Perceptrón simple.	7
	Visualizar la topología para el algoritmo Perceptrón simple.	
	Agregar el algoritmo para Redes Neuronales del Adaline.	
	Visualizar la ejecución y procesamiento del algoritmo Adaline.	
3	Visualizar la ejecución y procesamiento del algoritmo K-Means.	2
	Visualizar la ejecución y procesamiento del algoritmo K-NN.	

**Tabla 6. Plan de iteraciones.**

### 2.5.3 Plan de entrega

En el momento en que culmina la elaboración de las HU, se inicia el proceso de creación de un plan de entrega. El cual tiene como objetivo fundamental la obtención por parte de los programadores de una estimación detallada del período de tiempo que deben tener en cuenta para la implementación.

Artefacto	Iteración	Entrega
HU1	1	6 de febrero
HU2		
HU7		



HU3	2	4 de abril
HU4		
HU5		
HU6		
HU8		
HU9	3	21 de abril

**Tabla 7. Plan de entrega de las iteraciones.**

## **2.6 Fase de diseño**

### **2.6.1 Arquitectura de diseño**

Una arquitectura es el conjunto de decisiones significativas sobre la organización del sistema de software, la selección de los elementos y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplio y el estilo de arquitectura que guía esta organización, estos elementos y sus interfaces, sus colaboraciones y su composición (Jacobson, et al., 1999).

El WEKA presenta un diseño modular, con una arquitectura orientada a objeto que permite nuevos clasificadores, filtros, algoritmos de agrupamiento, para ser añadidos fácilmente. Además posee un conjunto de clases abstractas Java, una para cada tipo fundamental de componente, que fueron diseñadas y colocadas en su correspondiente nivel de paquetes. Los subpaquetes agrupan componentes de acuerdo a su funcionalidad o propósito. Por ejemplo, los filtros se dividen en aquellos que son supervisados o no supervisados, y posteriormente de acuerdo a si operan sobre un atributo o instancia básica. Los clasificadores se organizan de acuerdo al tipo general de algoritmo de aprendizaje, por ello existen subpaquetes para métodos Bayesianos, árboles inductivos y aprendices de reglas.

### **2.6.2 Patrones GRASP utilizados**

- **Bajo acoplamiento:** En la solución propuesta la relación entre las clases es la menor posible y en el caso de la herencia no está muy extendida ya que la idea es que existan la menor dependencia entre las clases, para de esta forma poder realizar modificaciones en alguna de ellas sin que impliquen cambios en las demás y potenciar la reutilización de las clases.

- **Alta cohesión:** Este patrón se utilizó en la solución con la idea de que cada clase dependa lo menos posible de otras para realizar su función principal.
- **Experto:** Este patrón se evidencia en las clases encargadas de realizar los diferentes algoritmos de Redes Neuronales como son el Adaline y SimplePerceptron.
- **Creador:** Este patrón se observa en las clases encargadas de crear las Redes Neuronales de los algoritmos Adaline y Perceptrón simple.
- **Controlador:** En la propuesta de solución este patrón se evidencia a la hora de controlar los datos del *dataset* que se envían a cada uno de los paquetes de la aplicación ya sea al panel de clasificación o agrupamiento.

### 2.6.3 Patrones GoF utilizados

- **Patrón Instancia única o Singleton:** En el software WEKA se evidencia el empleo de este patrón en el paquete de visualización, específicamente en la clase Main que es la encargada de controlar la creación de los diferentes paneles.

### 2.6.4 Tarjetas de Clase-Responsabilidad-Colaboración

Las tarjetas de Clase-Responsabilidad-Colaboración (CRC) permiten ver las clases no como un depósito de datos, sino que permiten conocer el comportamiento de cada una en un alto nivel. La metodología XP estipula su uso como un artefacto obligatorio durante el desarrollo de un proyecto, debido a los beneficios que aportan a los desarrolladores (Pressman, 2007). A continuación se muestran la tarjeta CRC de la clase Adaline. (Ver **Anexo 4: Tarjetas CRC**).

Nombre de la clase: Adaline	
Responsabilidades	Clases Relacionadas
Esta clase es la encargada de implementar el algoritmo de MD Adaline.	Classifier Instances Randomizable Instance Option Utils Filter

Tabla 8. Tarjeta CRC de la clase Adaline.

### **2.7 Conclusiones parciales**

En este capítulo se abordaron los aspectos fundamentales del análisis y diseño de la propuesta de solución. El levantamiento de los requerimientos del sistema permitió determinar las funcionalidades básicas a desarrollar durante el proceso. Se definieron 9 HU para implementarse en 3 iteraciones. Se identificaron las tarjetas CRC necesarias para la implementación de las nuevas funcionalidades, las cuales serán desarrolladas haciendo uso de los patrones de diseño.

## **CAPÍTULO 3: DESCRIPCIÓN DE LOS RESULTADOS**

En este capítulo se abordarán los aspectos fundamentales del proceso de desarrollo y pruebas. En el transcurso de todo este proceso se definirán el estándar de codificación y la implementación realizada al software WEKA. En la fase de pruebas se ejecutarán pruebas unitarias y de aceptación como forma de validación funcional de la propuesta de solución.

### **3.1 Estándar de codificación**

El estándar de codificación utilizado para el desarrollo de la solución propuesta, permite establecer una serie de reglas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los elementos fundamentales que componen el estándar utilizado se pueden encontrar en el **Anexo 5: Estándar de codificación**.

### **3.2 Fase de implementación**

La fase de implementación se inicia cuando se toma acción para alcanzar los resultados esperados. A continuación se describe las acciones y decisiones tomadas para la implementación de las nuevas funcionalidades que se le agregaron al software WEKA:

- **Preprocesamiento:** Esta clase es la encargada de cargar y procesar los datos con los que trabaja el software WEKA. En ella se muestran inicialmente unos histogramas para hacer una comparación entre todos los valores, pero su comprensión no era la más adecuada para entender los datos cargados. Esta situación provocó la necesidad de hacer una mejora visual con el fin de obtener un mejor análisis de los datos. Luego de un estudio se decidió diseñar e implementar un gráfico de pastel que cumpliera con esta tarea.
- **ID3:** Es un algoritmo de MD que se estudia en la asignatura de IA2 y mediante el software WEKA se analiza y explica su funcionamiento. Inicialmente el ID3 estaba implementado de tal forma que no mostraba el AD resultante de su corrida. Luego de un análisis se decidió pintar y mostrar el árbol resultante después de ejecutar dicha funcionalidad para así facilitar a estudiantes y profesores la comprensión del algoritmo, además de mostrar en consola paso a paso el proceso de selección de cada atributo para ir construyendo el árbol por niveles.
- **Perceptrón simple y Adaline:** Son algoritmos de Redes Neuronales que se estudian en la asignatura de IA2 y que no estaban implementados en el software WEKA. Para la incorporación de estos algoritmos se decidió implementar las clases SimplePerceptron y Adaline, encargadas conjuntamente con distintos métodos de

mostrar paso a paso, en consola, el funcionamiento y corrida de estos y visualizar la red neuronal que devuelven.

- **K-Means:** Es un algoritmo de agrupamiento que se estudia en la asignatura de IA2. Inicialmente K-Means solo mostraba la clasificación de los clustering en instancias. Por tanto se decidió mostrar los clustering pero con los valores que se les fue asignando, además mostrar una tabla con las instancias que se agrupan en cada clustering y los valores de esas instancias. Así se asegura que los profesores y estudiantes puedan observar de manera más organizada el funcionamiento de este algoritmo.
- **K-NN:** Es un algoritmo de clasificación que se estudia en la asignatura de IA2. Inicialmente K-NN solo mostraba la cantidad de instancias que se clasificaban bien y las que no. Por este motivo se decidió mostrar para cada una de las instancias como se seleccionaba un conjunto K de vecinos más cercanos con sus respectivas distancias y a partir de estos la nueva instancia es clasificada a la clase que pertenezcan la mayor cantidad de vecinos más cercanos.

### 3.3 Integración de las opciones de visualización a otras versiones de WEKA

Para integrar las nuevas opciones de visualización a futuras versiones del software WEKA es necesario incluir las librerías *jfreechart.jar* y *jcommon.jar* en el proyecto y reemplazar las siguientes clases en sus respectivos paquetes:

Clases	Paquetes	Subpaquetes
ClassifierPanel	GUI	Explorer
PintarRedAdaline	Classifier	Functions
PintarRed	Classifier	Functions
Adaline	Classifier	Functions
SimplePerceptron	Classifier	Functions
IBK	Classifier	Lazy
ID3	Classifier	Tree
SimpleKMeans	Clusteres	
AttributeVisualizationPanel	GUI	

Tabla 9. Clases a sustituir.

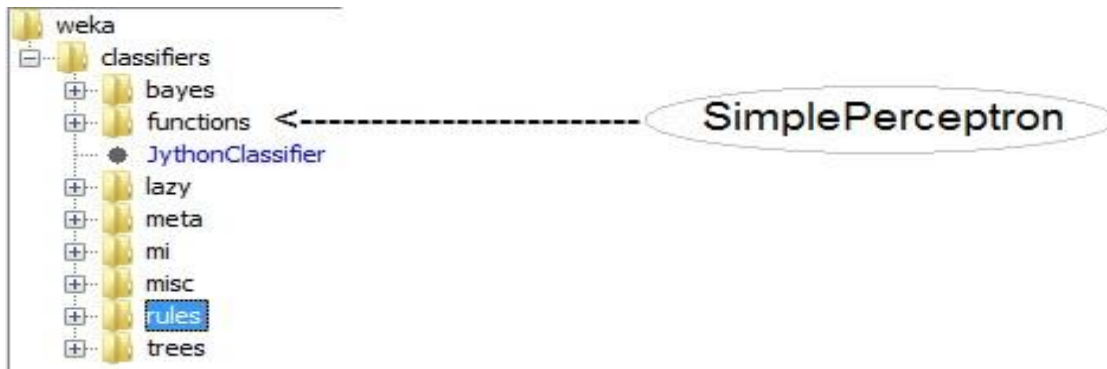


Figura 1. Ejemplo de donde sustituir las clases.

Luego de sustituir cada una de las clases y agregar las librerías se compila el proyecto para así poder hacer uso de las nuevas opciones de visualización de los algoritmos de AD ID3, K-Medias, K-Vecino más cercano y de los de Redes Neuronales Adaline y Perceptrón simple estudiados en clases y la nueva forma de mostrar los atributos mediante una gráfica de pastel.

### 3.4 Fase de pruebas

El objetivo fundamental de las pruebas es determinar si el código generado funciona correctamente y realiza las operaciones deseadas. Además permiten comprobar que el producto cumple con los requisitos. Para cumplir con lo antes mencionado se realizaron pruebas unitarias (encargadas de probar el código) y pruebas de aceptación (encargadas de probar los requisitos).

#### 3.4.1 Pruebas unitarias

Las pruebas unitarias se centran en un esfuerzo de verificación de la unidad más pequeña del diseño de software: el componente y el módulo del software. *“Las pruebas de unidad se centran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente”* (Pressman, 2007).

##### ➤ JUnit

Es un paquete de Java utilizado para automatizar los procesos de prueba. Mediante la creación de Test, realiza pruebas en el código. JUnit también es todo un entorno (framework<sup>15</sup>) para hacer pruebas de las clases Java de manera controlada, donde es posible manejar métodos por separado, especificar como debe ser la salida, verificar una condición y al final decidir si la prueba fue exitosa o no (JUnit, 2010). Existen varias razones para utilizar JUnit a la hora de hacer pruebas de código:

- ❖ La herramienta es gratuita.

<sup>15</sup> Framework: Marco de trabajo.

- ❖ Con JUnit, ejecutar pruebas es un proceso simple, basta con compilar el código. El compilador verifica la sintaxis del código y las pruebas "validan" la integridad del código.
- ❖ La propia aplicación chequea y muestra los resultados inmediatamente.
- ❖ Utilizando las pruebas programadas en JUnit, la estabilidad de las aplicaciones mejorará sustancialmente.
- ❖ Las pruebas realizadas se podrán presentar junto con el código, para validar el trabajo realizado.

➤ **Modo de empleo de JUnit:**

Como primer paso se vincula JUnit al IDE NetBeans y se selecciona la clase a la que se realizarán las pruebas. En este ejemplo para la clase Adaline, se creará automáticamente la clase AdalineTest, la cual contiene todas las funcionalidades de la clase a probar. Inicialmente en una primera instancia se muestra un listado con todos los errores detectados. Para las siguientes instancias de pruebas, mientras se le van insertando valores a los métodos de la clase AdalineTest se minimizan estos errores hasta llegar a un 100% de código limpio.

A continuación se muestran un grupo de imágenes con los resultados obtenidos de aplicar estas pruebas a la clase Adaline:

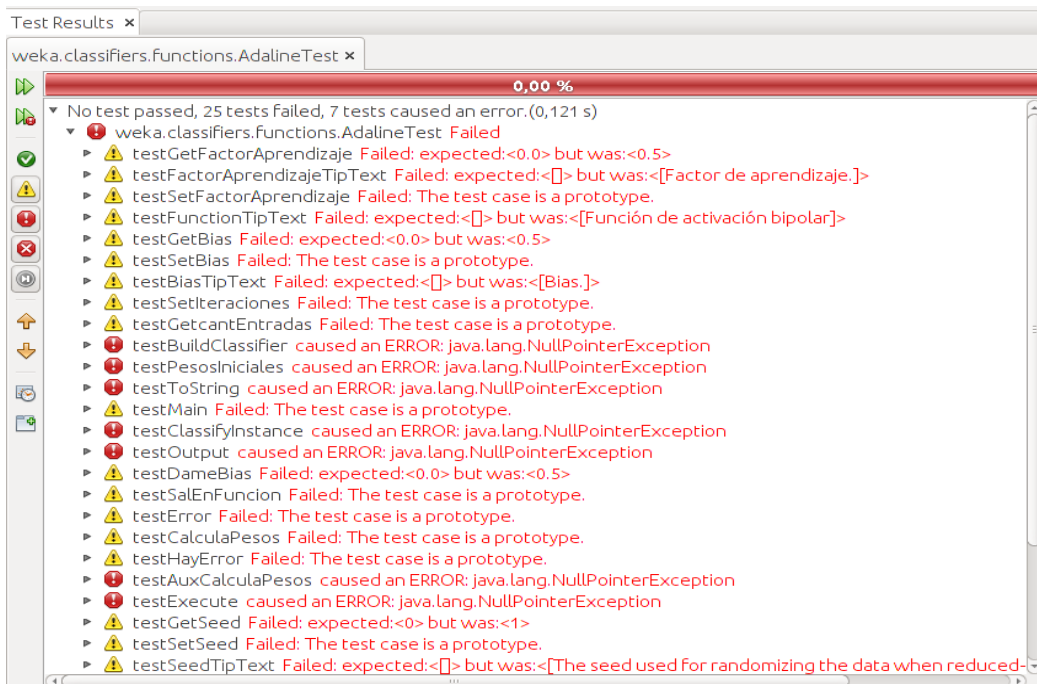


Figura 2. 1ra Instancia de la Prueba unitaria.

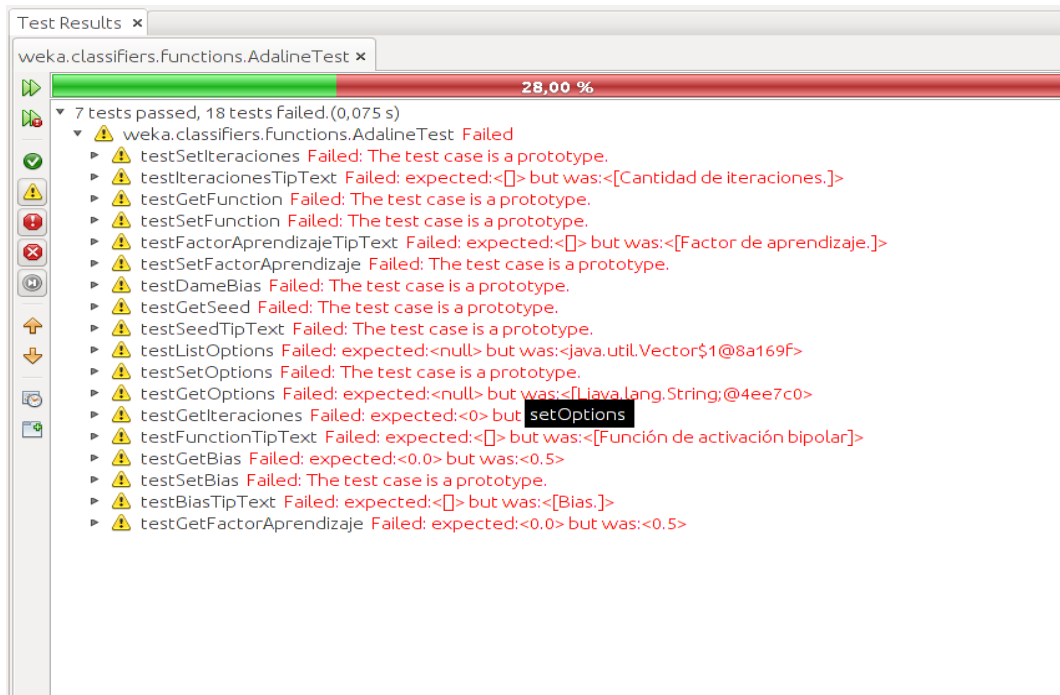


Figura 3. 2da Instancia de la Prueba unitaria.

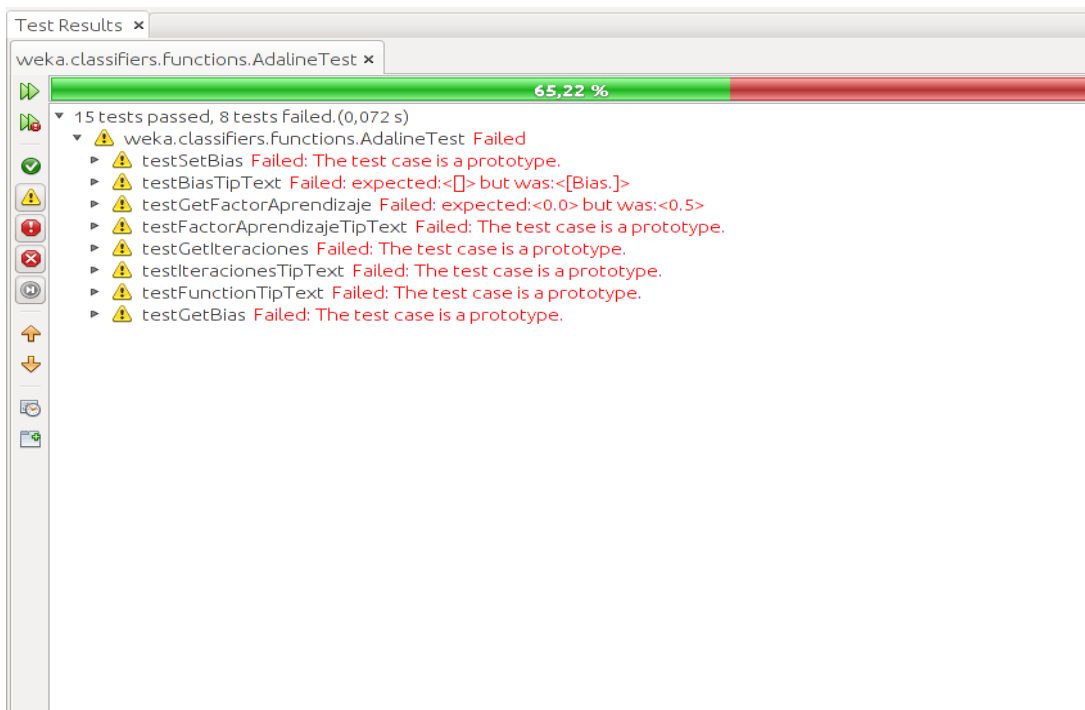
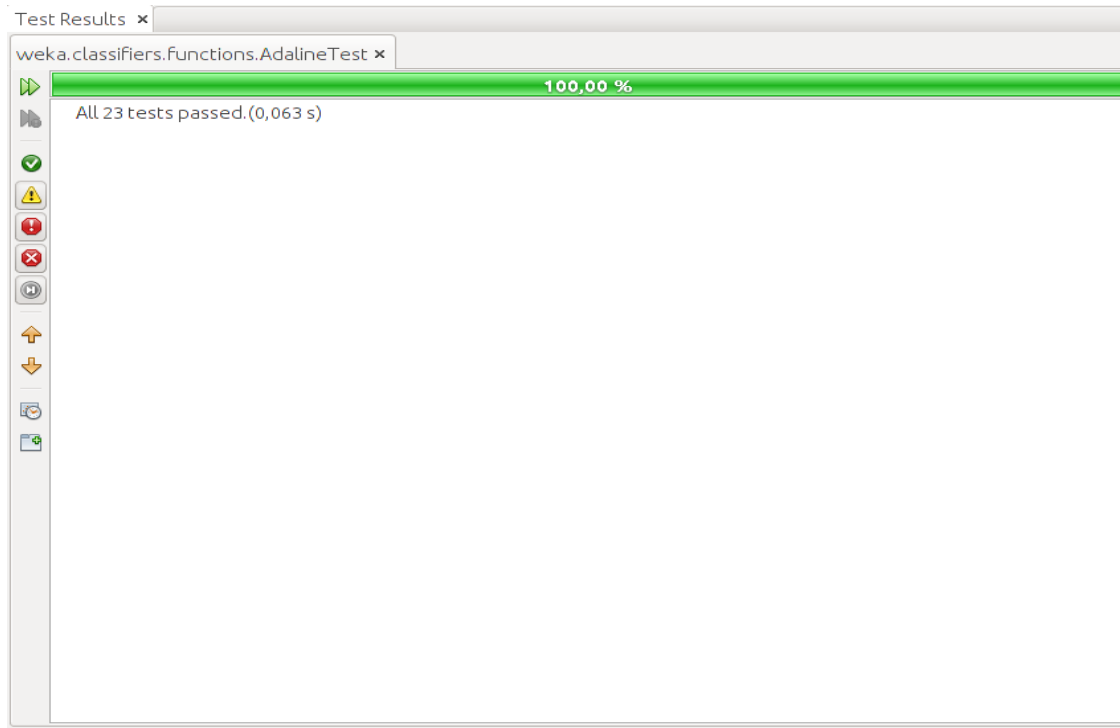


Figura 4. 3ra Instancia de la Prueba unitaria.





**Figura 5. 4ta Instancia de la Prueba unitaria.**

### **3.4.2 Pruebas de aceptación**

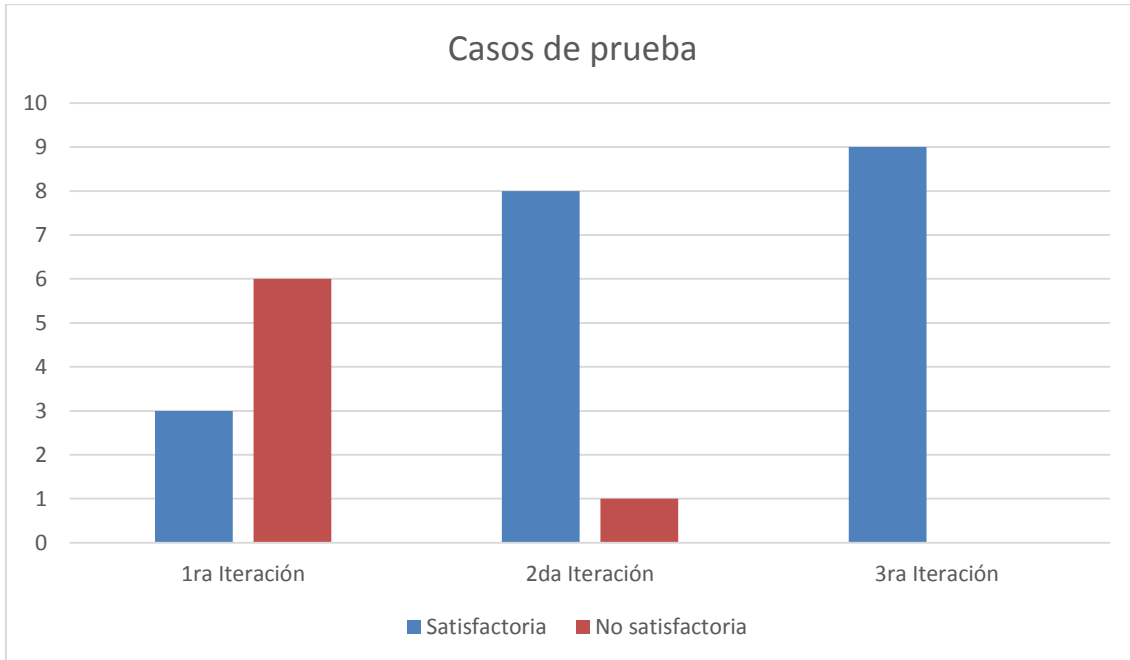
El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. Las pruebas de aceptación son una vía para comprobar esto. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique (Pressman, 2007).

➤ **Características de las Pruebas de aceptación:**

- ❖ Describe un escenario (secuencia de pasos) de ejecución o uso del sistema desde la perspectiva del cliente.
- ❖ Puede estar asociada a un requisito funcional o no funcional.
- ❖ Un requisito tiene una o más pruebas de aceptación asociadas.
- ❖ Las pruebas de aceptación cubren desde escenarios típicos/frecuentes hasta los más excepcionales.
- ❖ Una prueba de aceptación puede tener infinitas instancias (ejecuciones con valores concretos).

Para las Pruebas de aceptación se diseñaron 9 casos de prueba (Ver **Anexo 6: Casos de prueba**), de los cuales para una primera iteración se obtuvieron 3 de forma satisfactoria y 6 no satisfactoria. Para una segunda iteración se obtuvieron 8 casos de pruebas

satisfactorios y solamente 1 no satisfactorio. Y para una tercera iteración se obtuvieron los 9 casos de pruebas satisfactorios, resultados mostrados en la gráfica siguiente.



Las opciones de visualización de los algoritmos de análisis de datos agregados al software de MD WEKA, en la asignatura IA2, fueron aceptados y avalados poniéndose a disposición del: MSc Yuniesky Coca Bergolla, Jefe del Departamento de Técnicas de Programación y Sistemas Digitales de la Facultad 5 y Coordinador de la Disciplina de IA en la Universidad de las Ciencias Informáticas, reconociendo el cumplimiento de los objetivos planteados y el valor que brindan las nuevas funcionalidades al desarrollo de la asignatura IA2. En el **Anexo 7: Acta de validación** se presenta un aval que promueve su utilización.

### **3.5 Conclusiones parciales**

Como conclusiones del capítulo se abordaron las fases finales del proceso de desarrollo. Se estableció el estándar de codificación a utilizar lo que permitió determinar pautas para mejorar el desarrollo del código. En la fase de implementación se explicó los nuevos aportes realizados al software WEKA. En la fase de pruebas se examinó el código desde su parte más pequeña a través de las pruebas unitarias y se establecieron las pruebas de aceptación donde se comprobó que la implementación realizada se ajustaba a lo especificado en las HU. Además se validó que la solución pueda ser utilizada en las clases de IA2, mediante el Acta de validación.

## **CONCLUSIONES GENERALES**

La investigación desarrollada y los resultados obtenidos permitieron arribar a las siguientes conclusiones:

- La implementación de las clases SimplePerceptron y Adaline incorporó los algoritmos de Redes Neuronales Perceptrón simple y Adaline al software WEKA.
- Las nuevas opciones de visualización adicionadas al software WEKA proporcionaron a profesores y estudiantes una herramienta perfeccionada para el análisis de datos de los algoritmos que se estudian en la asignatura IA2.
- El resultado del trabajo realizado constituye un aporte para la herramienta que se utiliza en clases para impartir la asignatura Inteligencia Artificial 2.

## **RECOMENDACIONES**

A fin de enriquecer el trabajo realizado se recomienda:

- Agregar la implementación de la red neuronal McCulloch Pitts para operaciones con funciones lógicas.
- Mejorar la calidad de la imagen de las Redes Neuronales Adaline y Perceptrón simple.

## **BIBLIOGRAFÍA**

**2011.** ¿Qué es un entorno de desarrollo integrado? [En línea] 15 de Febrero de 2011. <http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/>.

*Adaline.*

**Alegsa, Leandro. 2010.** DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] 2010. <http://www.alegsa.com.ar/Dic/ide.php>.

**Bolivariana, Universidad. 2011.** *Programación Extrema.* 2011.

**Cabena, Peter, y otros. 1998.** *Discovering Data Mining From concept to implementation.* New Jersey : Prentice Hall, 1998. pág. 195.

**Centeno, Hender, y otros. 2011.** *Minería de Datos.* 2011.

**Chávez Solano, Ronald. 2010.** Modelo de Redes Neuronales. *Red Adaline.* [En línea] 21 de Noviembre de 2010. <http://ronaldchavezblog.blogspot.com/2010/11/red-adeline.html>.

**Cortés, Gloria y Casallas, Rubby. 2010.** *INTRODUCCIÓN A LOS PATRONES DE DISEÑO.* 2010.

**De Castro Riveras, Arturo Samuel. 2009.** NetBeans Programación. [En línea] 2009. <https://sites.google.com/site/netbeansprogramacion/news/anotherpost>.

**2008-2015.** Definición.de. [En línea] 2008-2015. <http://definicion.de/lenguaje-de-programacion/>.

**Estadística.** *Introducción a los árboles de decisión.*

**Galvis, Mario y Martínez, Fabricio. 2004.** *Confrontación de dos técnicas de Minería de Datos aplicadas a un dominio específico.* Facultad de Ingeniería. Bogotá, Colombia : s.n., 2004.

**Gamma, E., y otros. 1995.** *Desing Patterns.* s.l. : Adison Wesley, 1995.

**García Cambroner, Cristian y Gómez Moreno, Irene. 2006.** *Algoritmos de aprendizaje KNN y K-MEANS.* 2006.

**García Mondaray, Sergio. 2012.** Patrones de diseño GoF. [En línea] 2012. <http://www.godtic.com/blog/2012/11/15/patrones-de-diseno-gof/>.

**Gómez, Rosmary, Henríquez, Stefany y Obregón, Robert. 2013.** Inteligencia Artificial. [En línea] Febrero de 2013.

<http://inteligenciaartificialgrupo7.blogspot.com/2013/02/normal-0-21-false-false-false-es-ve-x.html>--inteligencia artificial.

**Gómez, Rosmary, Henríquez, Stefany y Obregón, Robert. 2013.** *Inteligencia Artificial. Áreas de Aplicación de la Inteligencia Artificial.* [En línea] 26 de febrero de 2013. <http://inteligenciaartificialgrupo7.blogspot.com/2013/02/normal-0-21-false-false-false-es-ve-x.html>.

**González Blanco, Rubén y Pérez, Sergio. 2010.** *Introducción a Rational Rose.* 2010.

**Gutiérrez, Damián. 2009.** Diagrama de Paquete. [En línea] 2009. [http://www.codecompiling.net/files/slides/UML\\_clase\\_05\\_UML\\_paquetes.pdf](http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf).

—. **2009.** *UML Diagramas de Paquetes (UML ilustrado).* 2009.

**Hernández Muñoz, José Luis. 2006.** Metodología de desarrollo de software. [En línea] 2006. <http://www.um.es/docencia/barzana/LAGP/lagp2.html>.

**Hernández Orallo, José y Ferri Ramírez, César . 2006.** *Introducción al Weka.* 2006.

**Jacobson, Ivar y Rumbaugh., Grady Booch y James. 1999.** *El Proceso Unificado de Desarrollo de Software.* . s.l. : Addison Wesley, 1999.

**2007.** *Java desde Cero.* 2007.

**JMACOE. 2011.** El rincón de JMACOE. *5 de los mejores software de minería de datos de Código Libre y Abierto.* [En línea] 2011. [http://blog.jmacoe.com/gestion\\_ti/base\\_de\\_datos/5-mejores-software-mineria-datos-codigo-libre-abierto/](http://blog.jmacoe.com/gestion_ti/base_de_datos/5-mejores-software-mineria-datos-codigo-libre-abierto/).

**JUnit. 2010.** JUnit. [En línea] 2010. <http://junit.org/>.

**Laguna, Miguel A. 2012.** *Requisitos.* 2012.

**López Martínez, Patricia y Aldea, Mario. 2011.** *Seminario de introducción al IDE Eclipse.* 2011.

**López Takeyas, Bruno. 2013.** *Algoritmo ID3.* 2013.

**López, Patricia. 2012.** *Herramienta CASE Visual Paradigm.* 2012.

**Maldonado, Daniel. 2007.** ¿Qué son los IDE de programación? [En línea] 2007. <http://elcodigok.blogspot.com/2007/09/que-son-los-ide-de-programacin.html>.

**Maneiro, M. 2008.** *Minería de Datos.* Buenos Aires, Argentina : s.n., 2008.

**Martínez, Mario Galvis Fabricio. 2004.** *Confrontación de dos técnicas de Minería de Datos aplicadas a un dominio específico.* Bogotá : s.n., 2004. pág. 114.

**Matich, Damian Joerge. 2011.** *Redes nueronales: Conceptos básicos y aplicaciones.* s.l. : 03, 2011.

**Molina López, José Manuel y García Herrero, Jesús. 2006.** *Técnicas de análisis de datos. Aplicaciones prácticas utilizando Microsoft Excel y Weka.* Madrid : s.n., 2006.

**Morate, Diego García. 2007.** *Manual del Weka.* 2007.

**2014.** NetBeans. [En línea] 9 de Enero de 2014.  
<http://www.genbetadev.com/herramientas/netbeans-1>.

**Pascual, D., Sánchez, S. y Pla, F. 2010.** *Algoritmos de agrupamiento.* Avenida Patricio Lumumba S/N 90500, Santiago de Cuba. Cuba : s.n., 2010.

**Peng, Wei, Chen, Juhua y Zhou, Haiping. 2012.** *An Implementation of ID3.* Sydney : s.n., 2012. pág. 20.

**Pressman, Roger S. 2007.** *Pressman.* 2007. Vol. VI.

—. **2007.** *Pressman.* 2007. Vols. IV-V.

**2012.** Redes de Neuronas Artificiales. *Perceptrón Simple y Adaline.* [En línea] 2012.  
<http://www.lcc.uma.es/~jmortiz/archivos/Tema4.pdf>.

**Rodríguez, Gilberto Velázquez. 2010.** *Diseño y aplicación de pruebas al producto Captura y Catalogación de Medias (CCM).* 2010.

**Sánchez, Jorge. 2004.** *Java2.* 2004.

**Sharkey, Noel. 2012.** BBC Mundo. *BBC Mundo, Tecnología.* [En línea] 5 de Junio de 2012.

[http://www.bbc.co.uk/mundo/noticias/2012/06/120621\\_turing\\_inteligencia\\_artificial\\_lp.shtml](http://www.bbc.co.uk/mundo/noticias/2012/06/120621_turing_inteligencia_artificial_lp.shtml).

**Spaerxs system. 2013.** Spaerxs system. *Spaerxs system.* [En línea] 2013.  
<http://www.sparxsystems.com.ar/>.

**2011.** Tecnología y Synergix. [En línea] 25 de Mayo de 2011.  
<http://synergix.wordpress.com.>

**2004.** *The ID3 selection Tree Algorithm.* 2004.

**Tomás Mariano, Víctor. 2011.** *Perceptrón I.* 2011.

**Vallejos, David Luis. 2006.** *Trabajo de Adscripción Minería de Datos.* Argentina : s.n., 2006.

**Zaragoza, María de Lourdes Santiago. 2013.** *Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado RUP.* 2013.



## **GLOSARIO DE TÉRMINOS**

**Algoritmo:** grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema. Método y notación en las distintas formas del cálculo.

**APIs:** Application Programming Interface: Interfaz de Programación de Aplicaciones.

Cluster: grupo

**Datos:** del latín datum ("lo que se da"), un dato es un documento, una información o un testimonio que permite llegar al conocimiento de algo o deducir las consecuencias legítimas de un hecho.

**Dataset:** conjunto de datos.

**Framework:** Marco de trabajo.

**GNU:** es un sistema operativo de tipo Unix desarrollado por y para el Proyecto GNU. GNU es un acrónimo recursivo de "GNU's Not Unix!" (En español: GNU no es Unix).

**GPL:** Licencia Pública General- General Public License.

**GRASP:** General Responsibility Assignment Software Patterns

**GoF:** Gang of Four, Pandilla de los Cuatro.

**GB:** Gigabyte, es una unidad de almacenamiento.

**HDD:** Unidad de Disco Duro: Hard Disk Drive.

**Java Development Kit:** Kit de desarrollo de Java.

**JBuilder:** es un Entorno de Desarrollo Integrado.

**K-NN:** K-Nearest Neighbor/Vecinos más cercanos

**Layout:** Disposición o plan.

**Minería:** el proceso de detectar y extraer objetos valiosos.

**PHP:** Hypertext Pre-Processor: Hipertexto Pre-procesador.

**PDF/HTML:** Portable Document Format/HyperText Markup Language.

**RAM:** Memoria de Acceso Aleatorio: Random Access Memory.

**Sistemas:** del latín systema, es un módulo ordenado de elementos que se encuentran interrelacionados y que interactúan entre sí.

**Tablets/smartphone:** dispositivo electrónico con múltiples aplicaciones.

**UML:** Lenguaje Unificado de Modelado (por sus siglas en inglés, Unified Modeling Language.)

**URL:** Localizador Uniforme de Recursos.

**Visualización:** es el acto y la consecuencia de visualizar. Este verbo, por su parte, refiere a desarrollar mentalmente la imagen de algo abstracto, a otorgar características visibles a aquello que no se ve.

**XOR:** es un tipo de disyunción lógica de dos operandos.

**XML:** Lenguaje de Marcas Extensibles.

**ANEXOS**

**Anexo 1: Patrones GoF**

Creacionales	Estructurales	Comportamiento
Método de fabricación	Adaptador	Intérprete
Fábrica abstracta	Puente	Método plantilla
Constructor virtual	Objeto compuesto	Cadena de responsabilidades
Prototipo	Decorador	Orden
Instancia única	Fachada	Iterador
	Peso ligero	Mediador
	Proxy	Recuerdo
		Observador
		Estado
		Estrategia
		Visitante

Figura 6. Patrones GoF.

**Anexo 2: Diagrama de paquetes**

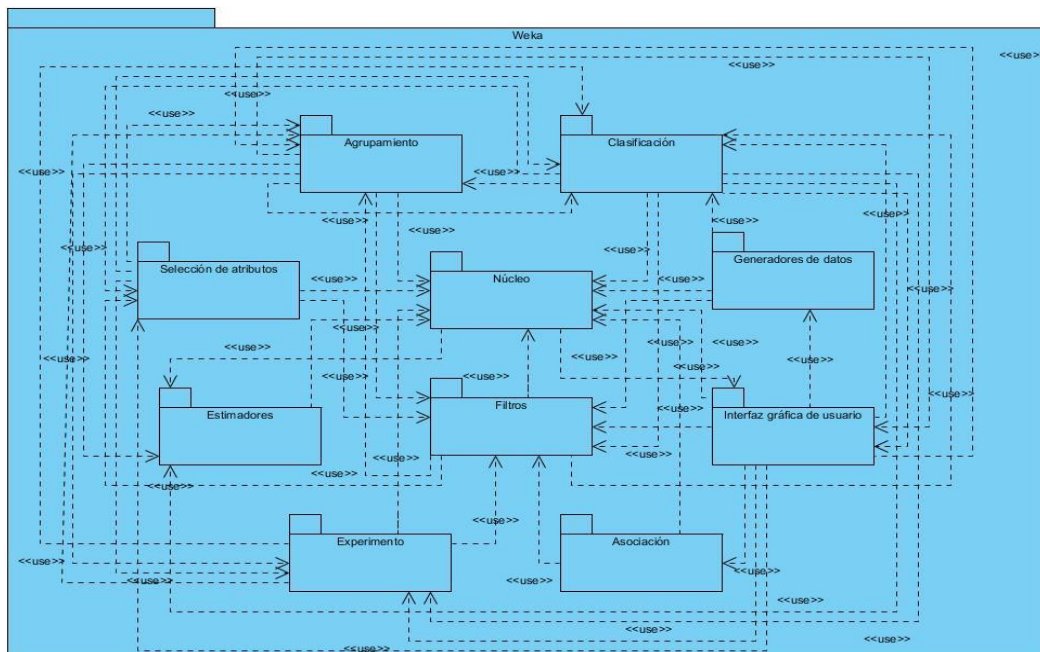


Figura 7. Diagrama de paquetes.

### Anexo 3: Historias de Usuario

Historia de Usuario	
<b>Número:</b> HU 2	<b>Nombre Historia de Usuario:</b> Visualizar el árbol de decisión del algoritmo ID3.
<b>Usuario:</b> Hayron Pérez	<b>Iteración Asignada:</b> 1
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 1
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar el árbol de decisión del algoritmo ID3.	
<b>Observaciones:</b>	

Tabla 10. Historia de usuario HU2.

Historia de Usuario	
<b>Número:</b> HU 3	<b>Nombre Historia de Usuario:</b> Agregar el algoritmo para Redes Neuronales del Perceptrón simple.
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 2
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 2
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 2
<b>Descripción:</b> La presente historia de usuario tiene como objetivo agregar el algoritmo para Redes Neuronales del Perceptrón simple.	
<b>Observaciones:</b>	

Tabla 11. Historia de usuario HU3.

Historia de Usuario	
<b>Número:</b> HU 4	<b>Nombre Historia de Usuario:</b> Agregar el algoritmo para Redes Neuronales del Adaline.
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 2
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 2
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 2
<b>Descripción:</b> La presente historia de usuario tiene como objetivo agregar el algoritmo para Redes Neuronales del Adaline.	
<b>Observaciones:</b>	

Tabla 12. Historia de usuario HU4.

Historia de Usuario	
<b>Número:</b> HU 5	<b>Nombre Historia de Usuario:</b> Visualizar la topología del algoritmo Adaline.
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 2
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 1
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar la ejecución y procesamiento del algoritmo Adaline.	
<b>Observaciones:</b>	

Tabla 13. Historia de usuario HU5.

Historia de Usuario	
<b>Número:</b> HU 6	<b>Nombre Historia de Usuario:</b> Visualizar la topología para el algoritmo Perceptrón simple.
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 2
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 2
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 2
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar la ejecución y procesamiento para el algoritmo Perceptrón simple.	
<b>Observaciones:</b>	

Tabla 14. Historia de usuario HU6.

Historia de Usuario	
<b>Número:</b> HU 7	<b>Nombre Historia de Usuario:</b> Visualizar la ejecución del algoritmo ID3.
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 1
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 1
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 1
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar la ejecución del algoritmo ID3.	
<b>Observaciones:</b>	

Tabla 15. Historia de usuario HU7.

Historia de Usuario
---------------------

<b>Número:</b> HU 8	<b>Nombre Historia de Usuario:</b> Visualizar la ejecución y procesamiento del algoritmo K-Means.	
<b>Usuario:</b> Hayron Pérez	<b>Iteración Asignada:</b> 3	
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 0.50	
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 1	
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar la ejecución y procesamiento del algoritmo K-Means.		
<b>Observaciones:</b>		

Tabla 16. Historia de usuario HU8.

Historia de Usuario		
<b>Número:</b> HU 9	<b>Nombre Historia de Usuario:</b> Visualizar la ejecución y procesamiento del algoritmo K-NN.	
<b>Usuario:</b> Handy Pereira	<b>Iteración Asignada:</b> 3	
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 1.50	
<b>Riesgo en Desarrollo:</b> Medio	<b>Puntos Reales:</b> 1.50	
<b>Descripción:</b> La presente historia de usuario tiene como objetivo visualizar la ejecución y procesamiento del algoritmo K-NN.		
<b>Observaciones:</b>		

Tabla 17. Historia de usuario HU 9.

## Anexo 4: Tarjetas CRC

Nombre de la clase: SimplePerceptron	
Responsabilidades	Clases Relacionadas
Esta clase es la encargada de implementar el algoritmo Perceptrón simple.	Classifier Instances Randomizable Instance Option Utils Filter

Tabla 18. Tarjeta CRC de la clase SimplePerceptron.

Nombre de la clase: ID3	
Responsabilidades	Clases Relacionadas
Esta clase es la encargada de implementar el algoritmo ID3.	Classifier Attribute Capabilities Instance Instances Drawable Utils

Tabla 19. Tarjeta CRC de la clase ID3.

Nombre de la clase: ClassifierPanel	
Responsabilidades	Clases Relacionadas
Esta clase es la encargada de mostrar e interactuar con el panel clasificador.	JPanel Classifier Attribute Capavilities Drawable Instance Instances Range Utils VisualizePanel TreeVisualize GraphVisualiza ExplorerPanel

Tabla 20. Tarjeta CRC de la clase ClassifierPanel.

Nombre de la clase: PreprocessPanel	
Responsabilidades	Clases Relacionadas

<p>Esta clase es la encargada de mostrar e interactuar con el panel de procesamiento de datos. Además de mostrar el gráfico de pastel y el histograma.</p>	<p>Explorer                  Capavilities                  Instances                  Utils                  Filter                  AttributeSelectionPanel                  AttributeSummaryPanel                  AttributeVisualizationPanel</p>
--	--

Tabla 21. Tarjeta CRC de la clase PreprocessPanel.

Nombre de la clase: ClustererPanel	
Responsabilidades	Clases Relacionadas
<p>Esta clase es la encargada de mostrar e interactuar con el panel de agrupamiento.</p>	<p>Explorer                  Attribute                  Capabilities                  Drawable                  Intance                  Intances                  Utils                  Filter                  VisualizePanel                  TreeVisualize</p>

Tabla 22. Tarjeta CRC de la clase ClusterPanel.

Nombre de la clase: SimpleKMeans	
Responsabilidades	Clases Relacionadas

<p>Esta clase es la encargada de implementar el algoritmo simple K-Means.</p>	<p>RandomizableClusterer</p> <p>Attribute</p> <p>Capabilities</p> <p>DistanceFunction</p> <p>EuclideanDistance</p> <p>ManhattanDistance</p> <p>Intance</p> <p>Intances</p> <p>Utils</p> <p>Filter</p>
---	---

Tabla 23. Tarjeta CRC de la clase SimpleKMeans.

Nombre de la clase: TreeVisualize	
Responsabilidades	Clases Relacionadas
<p>Esta clase es la encargada de implementar la visualización de un árbol.</p>	<p>PrintablePanel</p> <p>Instances</p> <p>Utils</p> <p>PrintablePanel</p> <p>VisualizePanel</p> <p>VisualizeUtils</p>

Tabla 24. Tarjeta CRC de la clase TreeVizualize.

Nombre de la clase: VisualizePanel	
Responsabilidades	Clases Relacionadas
<p>Esta clase es la encargada de implementar la visualización de un panel.</p>	<p>PrintablePanel</p> <p>Attribute</p> <p>FastVector</p> <p>Instance</p> <p>Instances</p> <p>ExtensionFileFilter</p> <p>Logger</p>

Tabla 25. Tarjeta CRC de la clase VizualizePanel.

Nombre de la clase: Attribute	
Responsabilidades	Clases Relacionadas



<p>Esta clase es la encargada de implementar el algoritmo Perceptrón simple.</p>	<p>Classifier Instances Randomizable Instance Option Utils Filter</p>
--	---

Tabla 26. Tarjeta CRC de la clase Attribute.

## Anexo 5: Estándares de codificación

### Nombres:

- Los nombres de las clases son sustantivos singulares.
- Los nombres deben reflejar el qué y no el cómo.
- Los nombres no deben revelar detalles de implementación.
- Evitar redundancia no repitiendo nombre de clases en sus elementos.
- Evitar la reutilización de nombres para distintos propósitos.

### Manejo de errores:

- Se pueden manejar errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.

### Documentación y Comentarios:

- Documentar mientras se programa.
- Al modificar el código se deben actualizar todos los comentarios y documentación asociada.
- Evitar agregar comentarios al final de líneas de código, salvo en el caso de declaraciones. En este caso, tales comentarios deben estar alineados.
- Antes de la entrega de la aplicación, eliminar todos los comentarios innecesarios y/o temporales con la finalidad de evitar confusiones en su mantenimiento.

### Codificación:

- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (declaraciones, lazos).
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.

**Anexo 6: Casos de prueba**


Caso de prueba de Aceptación	
<b>Código:</b> HU1_P1	<b>HU:</b> 1
<b>Nombre:</b> Gráfico de pastel.	
<b>Descripción:</b> El sistema debe mostrar una gráfica de pastel para mejorar la visualización en la etapa de preprocesamiento de datos.	
<b>Condiciones de ejecución:</b> El usuario debe cargar un conjunto de datos.	
<b>Entrada/Pasos de ejecución:</b> <ol style="list-style-type: none"> <li>1. Cargar los datos.</li> <li>2. Seleccionar en el combo box <sup>16</sup> de gráficos la opción Pastel.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar un gráfico de pastel, en la parte inferior derecha.	
<b>Resultado obtenido:</b>	
	
<b>Evaluación de la prueba:</b> Satisfactoria.	

Tabla 27. Caso de Prueba para Gráfico de Pastel.

Caso de prueba de Aceptación	
<b>Código:</b> HU2_P2	<b>HU:</b> 2
<b>Nombre:</b> Árbol de decisión ID3	
<b>Descripción:</b> El sistema debe mostrar la visualización del árbol de decisión del algoritmo ID3.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	

<sup>16</sup> Combo box: cuadro combinado.

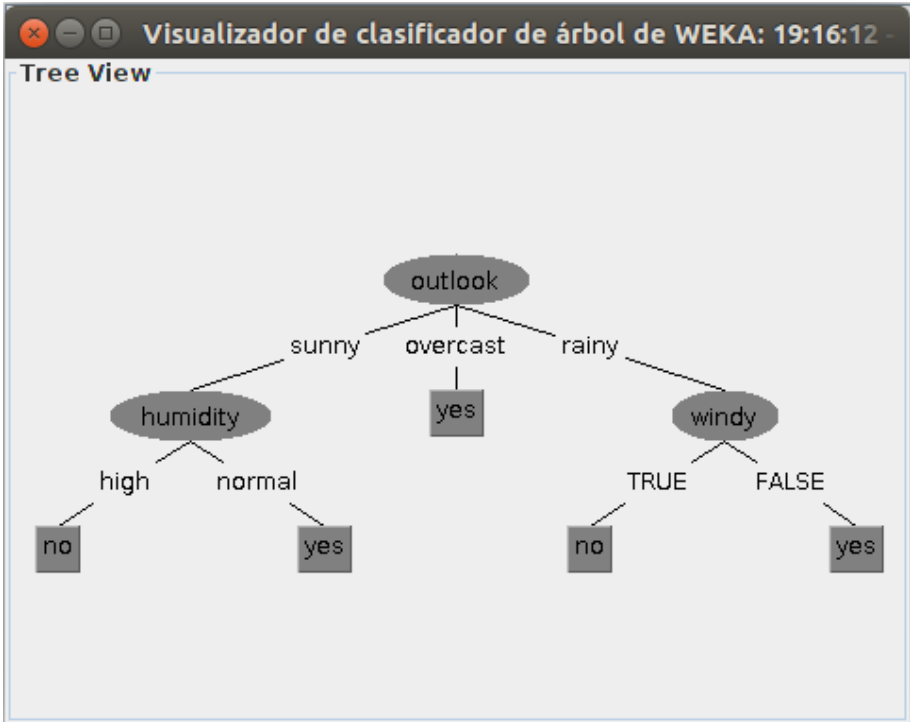
<p><b>Entrada/Pasos de ejecución:</b></p> <ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón Choose, y seleccionar el paquete de algoritmos Tree y en el mismo seleccionar ID3.</li> <li>3. Clic en el botón Iniciar ubicado en la parte izquierda.</li> <li>4. Después de ejecutar el algoritmo, clic derecho sobre la misma ejecución y seleccionar Visualizar árbol.</li> </ol>
<p><b>Resultado esperado:</b> El sistema debe mostrar en una nueva ventana el árbol de decisión ID3.</p>
<p><b>Resultado obtenido:</b></p>  <pre> graph TD     outlook([outlook]) -- sunny --&gt; humidity([humidity])     outlook -- overcast --&gt; yes1[yes]     outlook -- rainy --&gt; windy([windy])     humidity -- high --&gt; no1[no]     humidity -- normal --&gt; yes2[yes]     windy -- TRUE --&gt; no2[no]     windy -- FALSE --&gt; yes3[yes]     </pre>
<p><b>Evaluación de la prueba:</b> Satisfactoria.</p>

Tabla 28. Caso de prueba del AD ID3.

Caso de prueba de Aceptación	
<b>Código:</b> HU3_P3	<b>HU:</b> 3
<b>Nombre:</b> Algoritmo Perceptrón simple.	
<b>Descripción:</b> El sistema debe mostrar los resultados que devuelve el algoritmo de Redes Neuronales Perceptrón simple.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<p><b>Entrada/Pasos de ejecución:</b></p> <ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> </ol>	

2. Clic en el botón **Choose**, y seleccionar el paquete de algoritmos **Fuction** y en el mismo seleccionar **Perceptrón simple**.
3. Clic en el botón **Iniciar** ubicado en la parte izquierda.

**Resultado esperado:** El sistema debe mostrar los resultados del algoritmo de Redes Neuronales Perceptrón simple.

**Resultado obtenido:**

The screenshot shows the 'Opciones de prueba' (Test Options) panel on the left and the 'Salida del clasificador' (Classifier Output) panel on the right. The test options include 'Usar conjunto de entrenamiento', 'Suministrar conjunto de prueba', 'Validación cruzada' (selected), and 'Porcentaje dividido' (66%). The classifier output shows initial weights (w1-w4) and a table of results for 20 samples.

===Tabla de Resultados===												
X1	X2	X3	X4	Salida	En	F(En)	W1	W2	W3	W4	B	
0.0	0.0	0.0	1.0	1.0	0.709	1.0	0.048	0.344	0.54	0.209	0.5	
0.0	0.0	0.0	0.0	1.0	0.5	1.0	0.048	0.344	0.54	0.209	0.5	
1.0	0.0	0.0	1.0	0.0	0.758	1.0	0.048	0.344	0.54	0.209	0.5	
2.0	1.0	0.0	1.0	0.0	1.15	1.0	0.048	0.344	0.54	0.209	0.5	
2.0	2.0	1.0	1.0	0.0	2.034	1.0	0.048	0.344	0.54	0.209	0.5	
2.0	2.0	1.0	0.0	1.0	1.824	1.0	0.048	0.344	0.54	0.209	0.5	
1.0	2.0	1.0	0.0	0.0	1.776	1.0	0.048	0.344	0.54	0.209	0.5	
0.0	1.0	0.0	1.0	1.0	1.054	1.0	0.048	0.344	0.54	0.209	0.5	
0.0	2.0	1.0	1.0	0.0	1.937	1.0	0.048	0.344	0.54	0.209	0.5	
2.0	1.0	1.0	1.0	0.0	1.69	1.0	0.048	0.344	0.54	0.209	0.5	
0.0	1.0	1.0	0.0	0.0	1.384	1.0	0.048	0.344	0.54	0.209	0.5	
1.0	1.0	0.0	0.0	0.0	0.892	1.0	0.048	0.344	0.54	0.209	0.5	
1.0	0.0	1.0	1.0	0.0	1.297	1.0	0.048	0.344	0.54	0.209	0.5	
2.0	1.0	0.0	0.0	1.0	0.94	1.0	0.048	0.344	0.54	0.209	0.5	

**Evaluación de la prueba:** Satisfactoria.

Tabla 29. Caso de Prueba para Perceptrón simple.

Caso de prueba de Aceptación	
<b>Código:</b> HU4_P4	<b>HU:</b> 4
<b>Nombre:</b> Algoritmo Adaline.	
<b>Descripción:</b> El sistema debe mostrar los resultados que devuelve el algoritmo de Redes Neuronales Adaline.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<b>Entrada/Pasos de ejecución:</b> <ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Fuction</b> y en el mismo seleccionar <b>Adaline</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar los resultados del algoritmo de Redes Neuronales Adaline.	

**Resultado obtenido:**

**Opciones de prueba**

Usar conjunto de entrenamiento

Suministrar conjunto de prueba Set...

Validación cruzada Folds

Porcentaje dividido %

Más opciones...

(Nom) temperature

Iniciar Detener

Lista de resultados (pulse boton derecho del rató...)

19:10:02 - functions.Adaline

**Salida del clasificador**

Pesos iniciales:

w1=0.6610273122787476

w2=0.288976788520813

w3=0.4174656271934509

w4=0.726161777973175

===Historia de variables===

X1=outlook  
X2=temperature  
X3=humidity  
X4=windy  
Salida=play

===Tabla de Resultados===

X1	X2	X3	X4	Salida	En	F(En)	W1	W2	W3	W4	B
0.0	0.0	0.0	1.0	1.0	1.226	1.0	0.661	0.289	0.417	0.726	0.5
0.0	0.0	0.0	0.0	1.0	0.5	1.0	0.661	0.289	0.417	0.726	0.5
1.0	0.0	0.0	1.0	0.0	1.887	1.0	-0.283	0.289	0.417	0.254	0.5
2.0	1.0	0.0	1.0	0.0	0.478	1.0	-0.761	0.528	0.417	0.374	0.5
2.0	2.0	1.0	1.0	0.0	0.826	1.0	-1.587	1.354	0.004	0.167	0.5
2.0	2.0	1.0	0.0	1.0	0.039	1.0	-1.587	1.354	0.004	0.167	0.5
1.0	2.0	1.0	0.0	0.0	1.626	1.0	-2.4	0.541	0.411	0.167	0.5
0.0	1.0	0.0	1.0	1.0	1.209	1.0	-2.4	0.541	0.411	0.167	0.5
0.0	2.0	1.0	1.0	0.0	2.161	1.0	-2.4	-1.619	1.491	0.708	0.5
2.0	1.0	1.0	1.0	0.0	-3.72	0.0	-2.4	-1.619	1.491	0.708	0.5
0.0	1.0	1.0	0.0	0.0	0.372	1.0	-2.4	-1.805	1.398	0.708	0.5
1.0	1.0	0.0	0.0	0.0	-3.705	0.0	-2.4	-1.805	1.398	0.708	0.5
1.0	0.0	1.0	1.0	0.0	0.206	1.0	-2.503	-1.805	1.347	0.682	0.5
2.0	1.0	0.0	0.0	1.0	-6.311	0.0	4.808	-5.461	1.347	0.682	0.5

**Evaluación de la prueba:** Satisfactoria.

Tabla 30. Caso de Prueba para Adaline.

Caso de prueba de Aceptación	
<b>Código:</b> HU5_P5	<b>HU:</b> 5
<b>Nombre:</b> Red neuronal Perceptrón simple.	
<b>Descripción:</b> El sistema debe mostrar la red neuronal que devuelve el algoritmo de Perceptrón simple.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<b>Entrada/Pasos de ejecución:</b>	
<ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Fuction</b> y en el mismo seleccionar <b>Perceptrón simple</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> <li>4. Después de ejecutar el algoritmo, clic derecho sobre la misma ejecución y seleccionar <b>Visualizar red</b>.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar en una nueva ventana la red neuronal Perceptrón simple.	

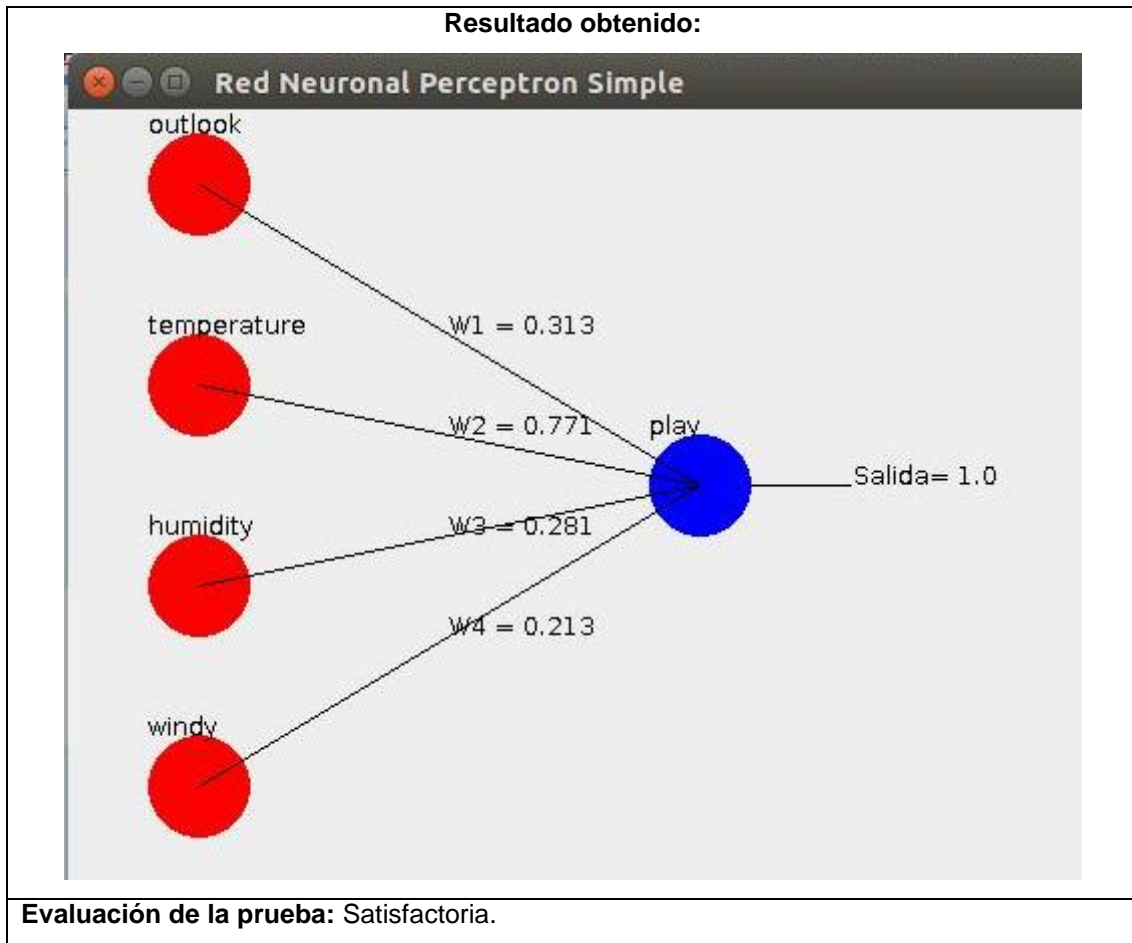
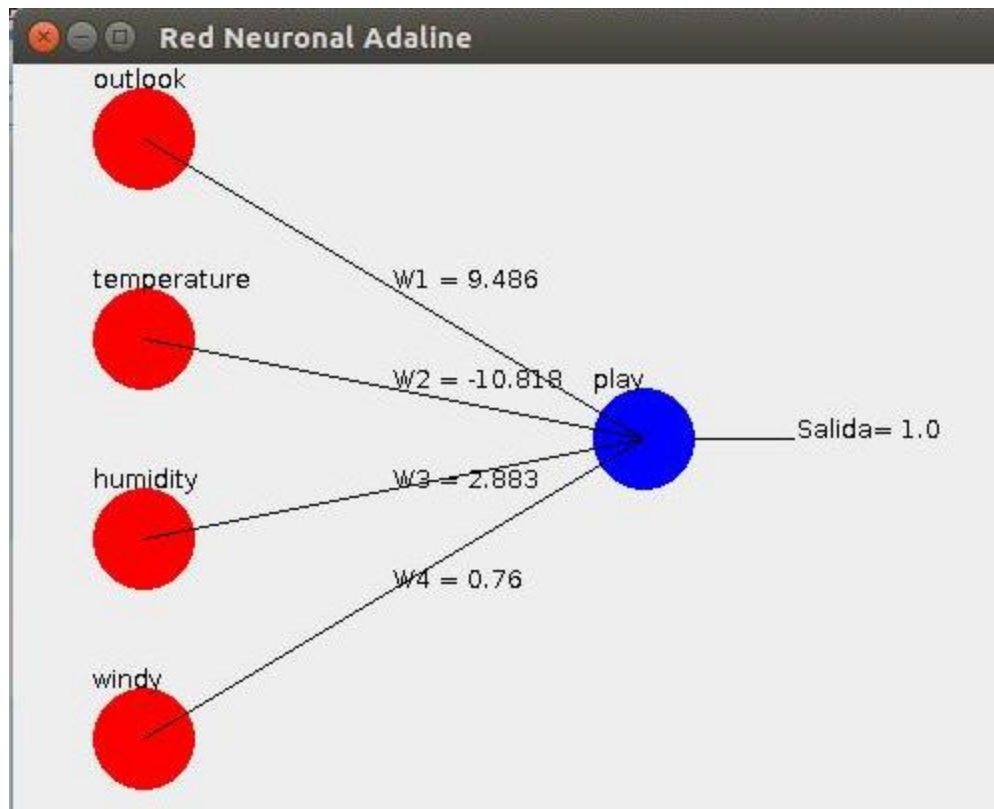


Tabla 31. Caso de Prueba para la red de Perceptrón simple.

Caso de prueba de Aceptación	
<b>Código:</b> HU6_P6	<b>HU:</b> 6
<b>Nombre:</b> Red neuronal Adaline.	
<b>Descripción:</b> El sistema debe mostrar la red neuronal que devuelve el algoritmo de Adaline.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<b>Entrada/Pasos de ejecución:</b>	
<ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Fuction</b> y en el mismo seleccionar <b>Adaline</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> <li>4. Después de ejecutar el algoritmo, clic derecho sobre la misma ejecución y seleccionar <b>Visualizar red</b>.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar en una nueva ventana la red neuronal Adaline.	

Resultado obtenido:



Evaluación de la prueba: Satisfactoria.

Tabla 32. Caso de Prueba para la red de Adaline.

Caso de prueba de Aceptación	
Código: HU7_P7	HU: 7
Nombre: Algoritmo ID3.	
Descripción: El sistema debe mostrar los resultados que devuelve el algoritmo ID3.	
Condiciones de ejecución: El conjunto de datos debe haber sido cargado.	
Entrada/Pasos de ejecución:	
<ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Tree</b> y en el mismo seleccionar <b>ID3</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> </ol>	
Resultado esperado: El sistema debe mostrar los resultados del algoritmo ID3.	

**Resultado obtenido:**

<p><b>Opciones de prueba</b></p> <p><input type="radio"/> Usar conjunto de entrenamiento</p> <p><input type="radio"/> Suministrar conjunto de prueba <span style="float: right;">Set...</span></p> <p><input checked="" type="radio"/> Validación cruzada <span style="float: right;">Folds <input type="text" value="10"/></span></p> <p><input type="radio"/> Porcentaje dividido <span style="float: right;">% <input type="text" value="66"/></span></p> <p style="text-align: center;">Más opciones...</p> <p>(Nom) play</p> <p style="text-align: center;">Iniciar <span style="margin-left: 50px;">Detener</span></p> <p>Lista de resultados (pulse boton derecho del rató...)</p> <p>19:16:12 - trees.Id3</p>	<p><b>Salida del clasificador</b></p> <p>Id3</p> <p>-----Iteraciones del algoritmo-----</p> <p>Atributo: outlook Entropia: 0.24674981977443894</p> <p>Atributo: temperature Entropia: 0.029222565658954536</p> <p>Atributo: humidity Entropia: 0.15183550136234125</p> <p>Atributo: windy Entropia: 0.04812703040826921</p> <p>Atributo: play Entropia: 0.0</p> <p>Atributo seleccionado es: Atributo: outlook Entropia: 0.24674981977443894</p> <p>Atributo: outlook Entropia: 0.0</p> <p>Atributo: temperature Entropia: 0.5709505944546683</p> <p>Atributo: humidity Entropia: 0.9709505944546684</p> <p>Atributo: windy Entropia: 0.019973094021974558</p> <p>Atributo: play Entropia: 0.0</p>
---	--

**Evaluación de la prueba: Satisfactoria.**

**Tabla 33. Caso de Prueba para el algoritmo ID3.**

Caso de prueba de Aceptación	
<b>Código:</b> HU8_P8	<b>HU:</b> 8
<b>Nombre:</b> Algoritmo K-Means.	
<b>Descripción:</b> El sistema debe mostrar los resultados que devuelve el algoritmo K-Means.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<p><b>Entrada/Pasos de ejecución:</b></p> <ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Cluster</b> y en el mismo seleccionar <b>K-Means</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar los resultados del algoritmo K-Means.	



**Resultado obtenido:**

**Modo de clúster**

**Uso de entrenamiento conjunto**

**Aparato de prueba suministrados** Conjunto...

**Porcentaje dividido** %

**Clases a grupos de evaluación**

(Nom) play

**Store clusters for visualization**

Ignorar los atributos

Iniciar
Detener

**Lista de resultados (clic en el botón derecho del rató...**

00:31:11 - SimpleKMeans

**Salida clusterer**

Valores de los Cluster

Cluster 0 = [sunny,mild,high,FALSE,yes]  
Cluster 1 = [overcast,cool,normal,TRUE,yes]

Cluster0	Instancias	Valores de las Instancias
=====		
	0	sunny,hot,high,FALSE,no
	1	sunny,hot,high,TRUE,no
	2	overcast,hot,high,FALSE,yes
	3	rainy,mild,high,FALSE,yes
	7	sunny,mild,high,FALSE,no
	8	sunny,cool,normal,FALSE,yes
	9	rainy,mild,normal,FALSE,yes
	10	sunny,mild,normal,TRUE,yes
	11	overcast,mild,high,TRUE,yes
	13	rainy,mild,high,TRUE,no

Cluster1	Instancias	Valores de las Instancias
=====		
	4	rainy,cool,normal,FALSE,yes
	5	rainy,cool,normal,TRUE,no
	6	overcast,cool,normal,TRUE,yes
	12	overcast,hot,normal,FALSE,yes

Clustered Instances

0	10 ( 71%)
1	4 ( 29%)

**Evaluación de la prueba:** Satisfactoria.

Tabla 34. Caso de Prueba para el algoritmo K-Means.

Caso de prueba de Aceptación	
<b>Código:</b> HU9_P9	<b>HU:</b> 9
<b>Nombre:</b> Algoritmo K-NN.	
<b>Descripción:</b> El sistema debe mostrar los resultados que devuelve el algoritmo K-NN.	
<b>Condiciones de ejecución:</b> El conjunto de datos debe haber sido cargado.	
<b>Entrada/Pasos de ejecución:</b> <ol style="list-style-type: none"> <li>1. El usuario debe seleccionar del panel de clasificación.</li> <li>2. Clic en el botón <b>Choose</b>, y seleccionar el paquete de algoritmos <b>Cluster</b> y en el mismo seleccionar <b>K-NN</b>.</li> <li>3. Clic en el botón <b>Iniciar</b> ubicado en la parte izquierda.</li> </ol>	
<b>Resultado esperado:</b> El sistema debe mostrar los resultados del algoritmo K-NN.	
<b>Resultado obtenido:</b>	

The screenshot shows the Weka Explorer interface. The 'Classifier' tab is active, displaying the following configuration and results:

**Classifier:** Choose | IBk -K 3 -W 0 -A "\weka.core.neighboursearch.LinearNNSearch -A "\weka.core.EuclideanDistance -R first-last"

**Opciones de prueba:**

- Usar conjunto de entrenamiento
- Suministrar conjunto de prueba (Set...)
- Validación cruzada (Folds: 10)
- Porcentaje dividido (%: 66)

**Lista de resultados (pulsar botón derecho del ratón):**

- 12:18:40 - lazy.IBK
- 12:18:52 - lazy.IBK

**Salida del clasificador:**

```

-----
Para la instancia #: 1
Con valores iguales a: sunny.hot.high.FALSE.no
-----
Vecino más cercanos #: 1
Valor del vecino más cercano : overcast.hot.high.FALSE.yes
Con una distancia de: 0.5

Vecino más cercanos #: 2
Valor del vecino más cercano : sunny.mild.high.FALSE.no
Con una distancia de: 0.5

Vecino más cercanos #: 3
Valor del vecino más cercano : sunny.hot.high.TRUE.no
Con una distancia de: 0.5

-----
Se clasifica en: no
-----

-----
Para la instancia #: 2
Con valores iguales a: sunny.hot.high.TRUE.no
-----
Vecino más cercanos #: 1
Valor del vecino más cercano : sunny.hot.high.FALSE.no
Con una distancia de: 0.5

Vecino más cercanos #: 2
Valor del vecino más cercano : sunny.mild.high.FALSE.no
Con una distancia de: 0.7071067811865476

Vecino más cercanos #: 3

```

**Evaluación de la prueba: Satisfactoria.**

Tabla 35. Caso de Prueba para el algoritmo K-NN.

## Anexo 7: Acta de validación

La Habana, Junio de 2015

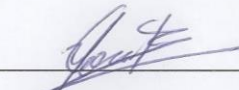
De mi consideración:

Por este medio se expresa la conformidad y aceptación de las opciones de visualización de los algoritmos del software de minería de datos Weka estudiados en la asignatura Inteligencia Artificial 2, a los cuales tributa el trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas, de los autores Hayron Yudiel Pérez Gómez y Handy Pereira Arencibia.

Las opciones de visualización incorporadas al software centralizan los principales problemas que presenta la herramienta Weka sobre todo para el aprendizaje de varios algoritmos tratados en clases. Dichas opciones permitirán a los estudiantes comprender mejor la minería de datos, específicamente mediante gráficas y esquemas de visualización que aportan a los algoritmos Árbol de Decisión (ID3), redes neuronales (Perceptrón simple y Adaline), así como a los algoritmos K-Means y K-NN.

Por lo antes mencionado se avala y aceptan las opciones de visualización incorporadas a los algoritmos de análisis de datos del Weka estudiados en Inteligencia Artificial 2, considerando su utilización para las clases de la asignatura antes mencionada.

Para que así conste se firma a los 3 días del mes de junio de 2015.



MSc. Yuniesky Coca Bergolla

Jefe del Departamento de Técnicas de Programación y Sistemas Digitales de la Facultad 5.  
Coordinador de la Disciplina Inteligencia Artificial en la Universidad de las Ciencias Informáticas.

Figura 8. Acta de validación.