Universidad de las Ciencias Informáticas
Facultad#6 " Bioinformática"

Título: "Módulo de predicción de la actividad biológica anticancerígena partiendo de descriptores topológicos e híbridos".

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor(es): Onailet Socorro Paz

Ileane Martí Pérez

Tutor(es): M.C. Aurelio Antelo Collado

Dr. Ramón Carrasco Velar

Consultante: Lic. Dannier Trinchet Almaguer

La Habana, Julio del 2007.
"Año 49 de la Revolución"

"Si tuviera el privilegio de vivir otra vez mi propia vida, muchas cosas las haría diferente de como las hice hasta hoy, pero puedo a la vez asegurar, que toda mi vida lucharía con idéntica pasión por los mismos objetivos por los que he luchado hasta hoy".

Fidel Castro.

DECLARACIÓN DE AUTORÍA

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

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

Ileane Martí Pérez

_____

Firma del autor

Onailet Socorro Paz

_____

Firma del autor

M.C. Aurelio Antelo Collado

_____

Firma del tutor

Dr. Ramón Carrasco Velar

_____

Firma del tutor

DATOS DE CONTACTO.

Tutores:

Dr.Ramón Carrasco Velar

Centro de Química Farmacéutica, Ciudad de La Habana, Cuba.

ramon.carrasco@cqf.sld.cu

M.C. Aurelio Antelo Collado

Universidad de las Ciencias Informáticas, Ciudad de LaHabana, Cuba.

aantelo@uci.cu

Consultante:

Lic. Dannier Trinchet Almaguer

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

trinchet@uci.cu

AGRADECIMIENTOS

A nuestros Padres, por el apoyo eterno y las ansias de vernos crecer.
A nuestros tutores, especialmente a Carrasco, por guiarnos en esta investigación.
A Yusdenis y Trinchet, por no fatigarse frente a nuestra incansable insistencia.
A los amigos, que han constituido fuente indispensable de sostén
y no habrá forma de olvidarlos nunca.
A los profesores, que durante la carrera nos enseñaron a aprender
y nos formaron como jóvenes merecedores de vivir en estos tiempos.
A la Revolución, por permitirnos formar parte del Destacamento Futuro,
fruto de la primera Universidad creada al calor de la Batalla de Ideas.
A la universidad más valiosa
y con más grandes expectativas de crear revolucionarios útiles e incondicionales,
por darnos la oportunidad de tener los mejores años de nuestras vidas.

Muchas Gracias a todos.

Es ahora que nos corresponde agradecerles con acciones, es ahora que retribuiremos con creces lo que con dedicación y espera lograron de nosotras.

DEDICATORIA

A mis padres, por ser la luz de mi vida, y estar apoyándome en todo momento.

A mi querida hermana, por ser quien es… un ejemplo a seguir.

A mis Abuelitas, por todo el amor que siempre me han dado y por haberme sabido guiar.

A Yusdenis Sánchez Perodín, por su amor e infinita ayuda.

A mis amigos, por ser incomparables e insustituibles.

Ileane Martí Pérez

A mami y papi que son lo más grande de este mundo.

A mi hermanita que es el brillo de mis ojos.

A mi abuelita Fide a cambio del tiempo que no he estado a su lado.

A Yine y familia, por ser mi propia familia durante estos años.

A Arles por hacer que estos últimos tiempos sean los mejores.

Y a mis amigos irremplazables.

Onailet Socorro Paz.

RESUMEN

El presente trabajo de investigación surge en el marco del proyecto conjunto entre el Centro de Química Farmacéutica y la Facultad 6 de la Universidad de las Ciencias Informáticas denominado “Una Plataforma Inteligente para la Predicción de Actividades Biológicas de Compuestos Orgánicos”. La obtención de la misma permitirá reducir el proceso de análisis y diseño de nuevas entidades químicas, las cuales servirán como propuesta para la obtención de nuevos principios activos, con lo cual se deben abaratar los costos de investigación y desarrollo de nuevos fármacos. Se analizó, diseñó e implementó una aplicación para la predicción de actividad biológica anticancerígena empleando lógica difusa como técnica de Inteligencia Artificial y se utilizaron índices topológicos e híbridos para describir las moléculas. Se realizaron las pruebas con dos muestras de datos reales. Una de ellas tomada del NCEA(National Center for Environmental Assessment) que sirvió para validar que el modelo desarrollado funcionaba correctamente y la otra del NCBI(National Center for Biotchnology Information). En esta última se dividió la muestra en muestras de entrenamiento (80%) y de prueba (20%). Los resultados en la clasificación fueron aceptables para esta primera versión de la aplicación pues se logró que el sistema clasificara correctamente el 79,1% de los compuestos, lo cual es atribuible a la calidad de los datos muestrales.

PALABRAS CLAVE

Lógica difusa, predicción, índices topológicos e híbridos.

TABLA DE CONTENIDO

INTRODUCCIÓN

El descubrimiento y desarrollo de un nuevo medicamento son dos fases que condicionan lograr que un nuevo producto sea útil en la terapéutica. Se plantea que el descubrimiento comprende toda la fase en la que se puede asegurar que el compuesto tiene un perfil deseable de actividad; comprende la síntesis, el aislamiento de la fuente natural, o la obtención biotecnológica y toda la fase preclínica, incluida la toxicología; de manera tal que se confirme que el compuesto es aceptable en cuanto eficacia y seguridad para su ensayo en seres humanos. Este largo proceso, comprende un total de 11.8 años de investigación, con un costo promedio de 231 millones de dólares por cada nuevo medicamento que salga al mercado. Lo más alarmante es que sólo una de cada 10 000 moléculas ensayadas pasa a la fase de desarrollo, una de cada 100 000 supera los ensayos clínicos y logra registrarse y sólo 3 de cada 10 nuevos medicamentos registrados recupera su inversión inicial. El desarrollo de medicamentos cada vez más seguros, adecuados y efectivos en el tratamiento de enfermedades, es una tarea que requiere del esfuerzo coordinado e inteligente de un elevado números de profesionales de distinta formación y dedicación, en la que la capacidad de deducción, la intuición y en muchos casos, la suerte, han jugado un papel fundamental(1) .

Una patología de elevada morbilidad y mortalidad es el cáncer, del cual se dispone de abundante información con respecto a entidades químicas evaluadas en diferentes ensayos. Todas estas razones la convierten en un blanco interesante y muy útil, desde el punto de vista informático para el establecimiento de relaciones entre la estructura química y la actividad biológica, razón por la cual se seleccionó la misma para realizar los estudios.

Por lo que puede decirse que el diseño racional de fármacos constituye una herramienta casi indispensable en el desarrollo actual de nuevos medicamentos, pues contribuye a un aumento de las posibilidades de éxitos y a una disminución tanto de los costos como de los tiempos de ejecución.

Entre las ramas que más han aportado a la solución de este problema se encuentran la Estadística y la Inteligencia Artificial. Todas estas técnicas de análisis de datos requieren de un manejo racional de la estructura química que brinde información automatizable. En la actualidad se utilizan distintas aplicaciones basadas en teoría de grafos, en las que se han implementado métodos para el cálculo de distintos descriptores estructurales, como por ejemplo el Codessa(2) , el Dragon(3), Molconn-Z(4), entre otros. Es

importante destacar que gran parte de los software que se usan con este fin son aplicaciones internacionales y/o comerciales, razón por la cual nuestros especialistas necesitan obligatoriamente de una conexión a Internet o de un fuerte financiamiento en divisas para poder utilizarlos. Teniendo en cuenta las dificultades financieras del país, de conexión a Internet, los problemas existentes con la brecha digital, y que por lo general estas aplicaciones no son gratuitas ni multiplataforma, se hace aún más necesario la creación de este módulo como parte del proyecto investigativo: "Una Plataforma Inteligente para la Predicción de Actividades Biológicas de Compuestos Orgánicos".

Una de las necesidades de la plataforma es conocer los resultados de predicción de la actividad anticancerígena utilizando diferentes técnicas de Inteligencia Artificial a partir de la descripción de la molécula por diferentes vías (fragmentos o descriptores). Entonces, ¿Cómo predecir la actividad anticancerígena para la Plataforma?

Para esto se tendrá como **objeto de estudio** la inteligencia artificial aplicada a la predicción de la actividad biológica, y como **campo de acción** la lógica difusa aplicada al estudio de la relación entre la estructura química y la actividad anticancerígena de compuestos orgánicos utilizando descriptores topológicos e híbridos.

Para obtener esta herramienta se plantea como **objetivo general** "desarrollar un módulo de predicción de la relación entre la estructura química y la actividad anticancerígena de compuestos orgánicos partiendo de descriptores topológicos e híbridos". Y como **objetivos específicos**:

- Analizar los módulos de generación de reglas y de predicción.
- Diseñar los módulos analizados.
- Implementar los módulos diseñados.
- Validar los módulos.

Para lograr los objetivos propuestos se trazaron las siguientes tareas:

- Estudio de los principales software utilizados en la predicción.
- Búsqueda de algoritmos para la generación de reglas difusas a partir de datos.
- Elaboración del modelo de dominio correspondiente.
- Definición de los requisitos funcionales y no funcionales.
- Desarrollo del diagrama de caso de uso del sistema.

- Descripción de la arquitectura.
- Desarrollo de los diagramas de clases del análisis.
- Desarrollo de los diagramas de clases del diseño.
- Desarrollo de los diagrama de iteración del diseño.
- Desarrollo del diagrama de despliegue.
- Desarrollo de diagrama de componentes.
- Validación del modelo.

Finalmente, el presente trabajo de diploma quedará estructurado con un resumen, introducción, cuatro capítulos fundamentales que constituyen el cuerpo del mismo, un apartado para las conclusiones generales y otro para las recomendaciones; así como para los anexos y el glosario de términos. A continuación se da un breve resumen de los capítulos.

Capítulo#1:"Fundamentación teórica". Discute las ideas básicas y las diferentes estrategias utilizadas en el aprendizaje y la clasificación. Se muestran algunos de los software de predicción existentes y se plantean las tecnologías, metodologías y software utilizados.

Capítulo#2:"Características del sistema". Se introduce con la definición del modelo de dominio. Se muestran los requisitos funcionales y no funcionales del sistema. Se describe el funcionamiento del sistema a través del diagrama de casos de uso del sistema y las descripciones de los casos de uso para comprender mejor el funcionamiento de la aplicación que se diseñará.

Capítulo#3:"Descripción de la solución". Se describe la arquitecrura utilizada,se realiza el análisis y el diseño de la aplicación que se desea obtener por medio de diagramas de clases del análisis y del diseño y además se presenta el diagrama de despliegue correspondiente.

Capítulo#4:"Implementación y Prueba". Comprende todo lo relacionado con la implementación del sistema mediante diagramas de componentes, y además todas las pruebas y validaciones correspondientes a la clasificación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se presenta el resultado de la revisión de métodos y algoritmos más adecuados para resolver el problema planteado. Se presentan además la metodología, el lenguaje de programación y el ambiente de desarrollo a utilizar como política de desarrollo del proyecto.

1.1 Introducción a la modelación molecular.

En el campo de los estudios de modelación molecular y diseño de fármacos es necesario describir la estructura química de manera cuantitativa para poder establecer los modelos que la correlacionen con los valores experimentales de actividad biológica. Precisamente estos valores cuantitativos son los llamados descriptores; los cuales pueden ser constitucionales, geométricos, electrostáticos, químico-teóricos, termodinámicos, topográficos, híbridos y topológicos. Estos dos últimos pueden emplearse para obtener las relaciones entre las propiedades químico-físicas, biológicas o ambas con la estructura química(5) por ejemplo, el índice de Wiener, los índices de conectividad molecular definidos por Randic, índice topológico de Hosoya, índice topológico de Schultz(6) ,los índices de Estrada(7) y los índices híbridos(8) entre otros.

En este trabajo se utilizarán descriptores topológicos e híbridos ya que estos se emplean ampliamente para el diseño de moléculas con propiedades deseadas y constituyen una potente herramienta para estos fines dada su facilidad de obtención y capacidad descriptiva de las moléculas o partes de ellas(5). Los descriptores utilizados son Randic, Valencia y Partición de la Refractividad, cada uno desde camino orden 1 - 15, cluster 3 y 4, además de sus combinaciones con los caminos desde 1-3 y además los ciclos. Por lo que cada una de esas combinaciones para cada índice representa una variable independiente, donde en total suman 72 variables de entrada.

Para poder determinar un modelo de predicción de estructura-actividad después de haberse seleccionado él o los descriptores que se emplearán, se pasa a establecer las diferentes relaciones existentes entre la estructura química y la actividad biológica a través de algún método estadístico u otras técnicas como las de Inteligencia Artificial. Una vez que la relación es generada sólo quedaría determinar la calidad del

modelo que puede realizarse usando métodos como el coeficiente de correlación, la validación cruzada o algún otro procedimiento.

La creación de modelos de predicción se han realizado usando sistemas de clasificación basados en árboles de regresión, máquinas de aprendizaje, análisis de discriminantes lineales, redes neuronales o lógica difusa. Este último tiene la ventaja de ser más condescendiente con datos imprecisos.

1.2 Técnicas y métodos utilizados en el diseño de fármacos.

Las relaciones cuantitativas entre parámetros fisicoquímicos y la actividad biológica mediante técnicas estadísticas las plantearon en los años 60 Hansch y Fujita. El ordenador empezó a ser necesario para resolver estas relaciones, denominadas QSAR (Quantitative Structure-Activity Relationships). Los métodos QSAR y QSPR (Quantitative Structure Property Relationship)(9) han demostrado que las relaciones entre la estructura molecular y las propiedades físico-químicas o actividades biológicas de los compuestos se pueden cuantificar matemáticamente a partir de parámetros estructurales simples. Más adelante, en la década de los setenta, los métodos de cristalización de proteínas y resolución de su estructura mediante difracción de rayos X, así como los de resonancia magnética nuclear, empezaron a aportar suficientes estructuras de macromoléculas como para pensar en un diseño de ligandos a partir del conocimiento tridimensional de su diana farmacológica

Durante las décadas de los 70 y 80 se produjo un gran desarrollo teórico de metodologías de modelación molecular y la aparición de los ordenadores personales permitió popularizar su aplicación. El penúltimo punto de inflexión en el desarrollo de fármacos tuvo lugar a mediados de los noventa, con la aparición de nuevas técnicas experimentales de química combinatoria. El empleo de estas técnicas abrió las puertas al desarrollo de metodologías computacionales para el diseño racional de bibliotecas de compuestos. Paralelamente, a ello se desarrolló explosivamente la informática, que permitió el procesamiento de enormes volúmenes de datos y la aplicación de la química cuántica, que si bien era conocida desde hace años, su aplicación práctica era inabordable con los medios de cómputo existentes(10). Actualmente, un

laboratorio moderno de diseño de fármacos debe contar con potentes medios de cómputo y de visualización de moléculas, que pueden ser propios o compartidos.

Hoy día la descripción de las moléculas no se realiza solamente con la utilización de propiedades químico-físicas determinadas experimentalmente. Son de uso común muchos de los descriptores mencionados anteriormente.

Actualmente se cuenta con aplicaciones que son de gran ayuda en este sentido para los especialistas, entre las que se encuentran: el Codessa (2)(Comprehensive Descriptors for Structural and Statistical Análisis), la cual brinda al usuario 116 descriptores de diferente naturaleza (constitucionales, topológicos, geométricos, electrostáticos, químico-teóricos, termodinámicos, etc.) y un módulo para el establecimiento de modelos QSAR basado fundamentalmente en análisis estadístico avanzado. Otra aplicación conocida es el Dragon(3), la misma calcula una gran número de descriptores, entre lo que se encuentran descriptores topológicos, índices de conectividad, descriptores geométricos, entre otros. Además está Molconn-Z(4), que calcula descriptores estructurales para su uso en métodos estadísticos con el fin de crear modelos QSAR para la predicción de la actividad biológica o propiedades físicas de nuevas moléculas. Todas estas aplicaciones generan una gran cantidad de información la cual no siempre se puede procesar utilizando técnicas estadísticas, es por eso la necesidad de utilizar otras técnicas.

En los últimos tiempos se han difundido y utilizado ampliamente algunas técnicas de Inteligencia Artificial en problemas de clasificación, en la predicción y de manera general en la toma de decisiones.

Dentro de la Inteligencia Artificial hay gran cantidad de técnicas que se pueden usar de forma independiente o combinar con otras, con el objetivo de encontrar mejores resultados, entre las que se encuentran: Redes Neuronales, Reconocimiento de Patrones, Máquinas de Soporte Vectorial, Algoritmos Genéticos, Lógica Difusa, y otras.

Se han realizado trabajos de predicción de las propiedades farmacológicas de compuestos moleculares basándose en sus características topológicas usando Redes Neuronales. Un ejemplo lo constituye(11) , donde los autores hacen un estudio del problema de la discriminación y predicción de las propiedades farmacológica de ciertos compuestos moleculares a partir de su topología utilizando perceptrones multicapa, como uno de los métodos de Redes Neuronales.

Por otro lado, para el desarrollo de modelos, relacionando las estructuras moleculares a su toxicidad y bioactividades, se han aplicado las Máquinas de Soporte Vectorial. Estas, experimentalmente han demostrado(12), que poseen capacidad predictiva comparable o superior a los modelos de Regresión Lineal Múltiple y de Redes de Funciones de Base Radial.

Hay dos ramas que se han destacado por sus aportes en el aprendizaje y la clasificación: los algoritmos de aprendizaje basados en árboles de decisión y los sistemas basados en el conocimiento. (13) . Las técnicas basadas en la lógica difusa forman parte de sistemas basados en el conocimiento. Estas técnicas han ganado gran popularidad específicamente los sistemas de inferencia borrosos y neuro-borrosos.

Un ejemplo lo constituye el trabajo realizado por (14). Donde se demuestra cómo los clasificadores borrosos se pueden utilizar para generar relaciones borrosas de la estructura-actividad.

Entre otras aplicaciones para la predicción de actividad biológica se conoce por ejemplo el sistema experto APEX-3D(15) implementado sobre Silicon Graphics que utiliza teoría lógico combinatoria para el establecimiento de las relaciones estructura-actividad de grupos de compuestos que constituyen la base de datos suministrada por el usuario. Este sistema experto está insertado dentro del paquete de programas Insight II y tuvo su predecesor en el sistema experto OREX implementado sobre IBM PC 80286. Este último se basa en la descomposición topológica de las moléculas en fragmentos estructurales y su asociación a las actividades biológicas reportadas en una base de datos interna de alrededor de 15 000 compuestos. Emplea también la teoría lógico-combinatoria para el establecimiento de reglas de inferencia.

El programa MCASE(16), es creado para computadoras que tengan los sistemas operativos Windows NT/2000/XP. Acepta la estructura de una serie de compuestos con sus respectivas actividades y realiza correlaciones QSAR con las cuales crea diccionarios que son usados para predecir las actividades de compuestos desconocidos. Cuando se introduce un compuesto nuevo lo evalúa contra los diccionarios creados, calcula el QSAR apropiado y realiza una predicción de acuerdo a la actividad sugerida.

Otro software es OncoLogic (17), el cual ha sido desarrollado cooperativamente entre la EPA's Office of Pollution Prevention and Toxics (OPPT) y LogiCheminc, sobre PC tipo IBM-compatible DOS (no-

Windows). Analiza las estructuras químicas para determinar la probabilidad de provocar cáncer mediante la aplicación de reglas mediante análisis SAR (Structure-Activity Relationships), y la incorporación de conocimientos sobre cómo las sustancias químicas causan cáncer en animales y humanos. Es el único sistema experto para predicción de cancinogenicidad que evalúa no sólo la estructura química sino también factores no estructurales, propiedades físicas y la ruta de la exposición. Aunque hace uso de ciertos datos de la característica física dondequiera que esté disponible, no utiliza los modelos QSAR en sus evaluaciones, y no puede calcular características fisicoquímicas para apoyar tales modelos.

Otro sistema conocido es el ADAPT(18) (Automated Data Analysis using Pattern Recognition Toolkit). Este es un sistema de programas que le permite al usuario el desarrollo de relaciones estructura-actividad y estructura-propiedad. Brinda al usuario la facilidad de entrada gráfica y almacenamiento de estructuras moleculares y sus datos asociados, generación de estructuras 3D, cálculo de descriptores moleculares y análisis de estos empleando estadística multivariada, reconocimiento de patrones o redes de neuronas para construir modelos predictivos. Posee una gran selección de rutinas generadoras de descriptores moleculares (topológicos, geométricos, electrónicos y físico-químicos). Los enfoques estadísticos incluyen regresión lineal múltiple, análisis clúster, discriminante y redes neuronales. Se ejecuta sobre estaciones de trabajo Sun bajo sistema operativo UNIX.

A pesar de que estas aplicaciones predicen, no resuelven el problema planteado, ya que se pretende establecer una relación entre los descriptores topológicos e híbridos con la actividad biológica usando la lógica difusa como técnica de inteligencia artificial, y se incorporarán al estudio los descriptores híbridos, los que han sido desarrollados por especialistas cubanos.

1.3 ¿Qué es la Lógica Difusa?

La lógica difusa o borrosa, es la lógica que utiliza expresiones que no son ni totalmente ciertas ni completamente falsas, es la lógica aplicada a conceptos que pueden tomar un valor cualquiera de veracidad dentro de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total. Esta permite representar el conocimiento común, que es mayoritariamente del tipo lingüístico cualitativo y no necesariamente cuantitativo, en un lenguaje matemático a través de la teoría de

conjuntos difusos y funciones características asociadas a ellos. Permite trabajar a la vez con datos numéricos y términos lingüísticos. Estos últimos son inherentemente menos precisos que los datos numéricos pero en muchas ocasiones aportan una información más útil para el razonamiento humano.

Una de las ventajas de la lógica difusa es la posibilidad de implementar sistemas tanto en hardware como en software o en combinación de ambos, y que se requiere un bajo costo computacional y tienen baja sensibilidad al ruido. En la actualidad es un campo de investigación muy importante, tanto por sus implicaciones matemáticas o teóricas como por sus aplicaciones prácticas. En general la lógica difusa se aplica a sistemas de control y para modelar cualquier sistema continuo de Ingeniería, Física, Biología o Economía. Las aplicaciones van desde procesos tan complejos como control de sistemas, ej. Control de tráfico etc.; predicción y optimización. ej. predicción de terremotos, optimización de horarios, etc.(19) ; hasta cosas tan cercanas a todos como son las lavadoras que tienen control automático del agua, el detergente y el desagüe, y que pueden identificarse claramente porque dicen "Fuzzy Logic". Este es quizás una de las aplicaciones más conocidas y populares de las técnicas de Inteligencia Artificial (20).

1.3.1 Etapas de un sistema lógico difuso.

Un sistema lógico difuso de manera general se basa en tres etapas (Figura #1):

a. Fusificación. Término anglófono que se utiliza para convertir los valores nítidos en valores borrosos o difusos.

b. Reglas de Evaluación o Inferencia Difusa como también se conocen.

c. Defusificación. Término anglófono que se utiliza para convertir los valores difusos obtenidos en el proceso de inferencia, en valores nítidos.

En los siguientes epígrafes se explica en que consiste cada una de ellas.

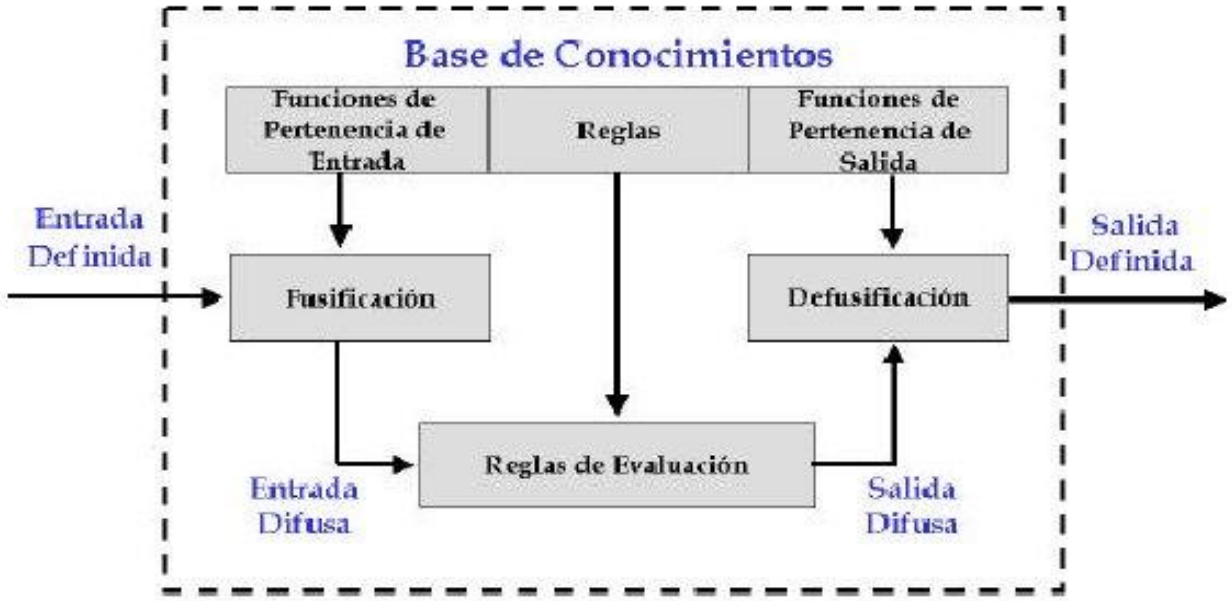


Figura #1 Representación gráfica de las etapas de un sistema lógico difuso.

Fusificación

Los elementos fundamentales en esta etapa son las Funciones de Pertenencia de Entrada. Existen gran cantidad de funciones de este tipo entre las que podrían citarse:

- Función de pertencia puntual.
- Función Gamma: Escalones crecientes.
- Función L: Escalones decrecientes.
- Función LAMBDA: Funciones triangulares.
- Función Pi: Funciones trapezoidales.

En el presente trabajo se utilizó la función de tipo trapezoidal, también conocido como Pi (Ver Anexo#1) y la cantidad de conjuntos difusos definidos para cada variable fueron cinco. La variable del proceso (entrada definida, no-difusa o crisp) intercepta las Funciones de pertenencia generando las entradas difusas. Mediante este procedimiento, el fusificador establece una relación entre los puntos de entrada no difusos y sus correspondientes conjuntos difusos en el universo de discurso U.

Reglas de Evaluación

Las Reglas son sentencias SI-ENTONCES que describen las condiciones (antecedentes) y las acciones (consecuentes) que deben existir para tomar una decisión. La sintaxis de las reglas es la siguiente:

SI Antecedente 1 Y Antecedente 2. . . ENTONCES Consecuente 1 Y. . .

El antecedente de una regla puede tener muchas partes, en tal caso todas las partes del antecedente son calculadas simultáneamente y transformadas en un número utilizando los operadores lógicos (Ver Anexo#2). El consecuente de una regla puede también tener partes múltiples y todos los consecuentes son afectados de la misma manera por el antecedente.

Las reglas difusas permiten expresar el conocimiento que se tiene acerca de la relación entre los antecedentes y los consecuentes en un cierto grado de verdad. Para expresar este conocimiento de forma completa normalmente se precisa de varias reglas, con pesos asociados, que se agrupan formando una base o bloque de reglas.

Mediante la inferencia los sistemas difusos interpretan las reglas de tipo SI-ENTONCES contenidas en su base de conocimientos, con el fin de obtener los valores de salida a partir de los valores que tienen las variables lingüísticas de entrada al sistema. Una vez que las entradas crisp han sido convertidas a variables de valores lingüísticos (Fusificación), se utiliza la inferencia difusa para identificar las reglas de tipo SI-ENTONCES que se aplican a la situación actual y se calculan los valores lingüísticos de salida.

Defusificación.

La salida del proceso de inferencia es hasta ahora un conjunto difuso que indica la posibilidad de realizar una acción de control. Sin embargo, las aplicaciones de los sistemas difusos no pueden interpretar los valores lingüísticos obtenidos, por lo que las funciones de pertenencia de salida son utilizadas para retransformar los valores difusos nuevamente en valores definidos o crisp mediante la defusificación (21).
Algunas de las funciones utilizadas para realizar este proceso son:

- Centro de área.
- Media difusa ponderada.
- Máximo extremo izquierdo.
- Máximo extremo derecho .
- Centro de gravedad.

Puede decirse que el centro de gravedad y el centro de área son métodos equivalentes. El método de defusificación utilizado en el presente trabajo de diploma es el Centro de Gravedad (COG siglas en ingles o centroide, ecuacion#1), el cual es uno de los métodos más usado en las técnicas de defusificación. Lo interesante de este método es que no hay que ajustar ningún coeficiente, sino que lo necesario es simplemente conocer las funciones de pertenencia de cada una de las etiquetas definidas.
El método COG parte de los valores de pertenencia a cada una de las etiquetas asociadas a la variable que se quiere defusificar.

$$COG = \frac{\int M(x) * x dx}{\int M(x) dx}$$

Ecuación#1:Método COG

1.3.2 Modelo de Mamdani. Principales características.

La forma de cómo resolver un problema mediante lógica difusa es muy variada, esta no es una técnica rígida y en dependencia de las características del problema será la forma de proceder. Hoy se cuenta con diferentes metodologías y arquitecturas entre las que están FIR(Fuzzy Inductive Reasoning)(22) y ANFIS(Adaptive Neuro Fuzzy Inference System)(23) respectivamente, las que combinan la lógica difusa con otras técnicas dígase algoritmos evolutivos, redes neuronales, etc. Además existen tres modelos para los sistemas basados en la lógica difusa: Mamdani, Sugeno el cual también es conocido como Takagi, Kang o TSK y el modelo Tsukamoto. Los cuales tienen características específicas que lo hacen también ser usado en una implementación dependiendo del problema a resolver. La principal diferencia entre los modelos es en las consecuencias de las reglas y en los métodos de agregación y defusificación(24). En el presente trabajo se utilizará el modelo de Mamdani, cuya principal característica esta dado en las reglas SI-ENTONCES.

En un Sistema Difuso tipo Mamdani tanto el antecedente como el consecuente de las reglas están dados por expresiones lingüísticas. Los sistemas basados en reglas difusas tipo Mamdani poseen las siguientes características:

Ventajas:

- Facilidad para la derivación de reglas.
- Interpretabilidad de las reglas difusas.
- Fueron propuestos antes y se han utilizado con más frecuencia.

Inconvenientes:

- No garantizan la continuidad de la superficie de salida.
- Menor eficiencia computacional.

Estas reglas difusas se pueden obtener de diversas formas, mediante el conocimiento de un especialista, lo cual, cuando no es suficiente, hay que encontrar algoritmos o técnicas que logren generar dichas reglas a partir de datos de entrada.

Actualmente existen sistemas que han logrado extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos, por ejemplo:

ANFIS: El cual fue anteriormente mencionado**,** permite sintonizar o crear la base de reglas de un sistema difuso, utilizando el algoritmo de entrenamiento de retropropagación a partir de la recopilación de datos de un proceso, pero su arquitectura es funcionalmente equivalente a una base de reglas tipo Sugeno.

FSOM (Fuzzy Self-Organizing Maps): Consiste en un sistema difuso optimizado a partir de los mapas auto-organizados de Kohonen.

Fuzzy Tech: Es un software que propone un método de desarrollo de sistemas neuro-difuso similar a ANFIS.

MLRUL y GENRUL5: Son algoritmos que permiten la generación de reglas borrosas a partir de datos numéricos y simbólicos El algoritmo MLRUL posibilita además la manipulación de rasgos irrelevantes, la ausencia de información y manejo de datos con información ruidosa. Estos algoritmos permiten el tratamiento de casos con información incompleta, pero la estructura de las reglas generadas son tipo Sugeno grado cero(13), (25).

NefClass: Este algoritmo está basado en la estructura del perceptrón multicapa cuyos pesos son modelados por conjuntos difusos. Así, se preserva la estructura de una red neuronal, pero se permite la interpretación del sistema resultante por el sistema difuso asociado. La estructura que poseen las reglas difusas son tipo Mamdani(26). NefClass es el algoritmo que se utilizará en esta tesis para obtener las reglas difusas, donde se generan automáticamente la mayor catidad de reglas, con criterio de selección “mejores reglas por clase”,pudiendo realizarse una poda de las reglas posteriormente a su creación o simultáneamente a la creación de estas.

1.4 Tendencias y tecnologías actuales.

Para la realización de un software siempre se debe definir los tipos de tecnologías a utilizar así como las herramientas que serán de mayor utilidad para su implementación. A continuación se explica en detalle cada una de las tecnologías que fueron seleccionadas para llevar a cabo la implementación y documentación del software en cuestión.

1.4.1 Metodologías de desarrollo.

La dificultad que presentan hoy en día los desarrolladores a la hora de realizar un software es la necesidad de saber cómo organizar las actividades para cada desarrollador por separado y para el equipo, definir qué artefactos deben ser creados y contar con una serie de criterios que permitan controlar y medir los productos que se obtienen, por lo tanto se necesita de una metodología capaz de dirigir estas actividades y así convertir los requisitos de los usuarios en un producto software. En el mundo existen distintas metodologías para dirigir las actividades vinculadas al proceso de desarrollo de software, entre estas metodologías se estudiaron las siguientes:

XP: Extreme Programing

Una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo. XP fue inicialmente creada para el desarrollo de aplicaciones dónde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación.

XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

Puntos claves:

- Ligero.
- Cercano al desarrollo.
- Se basa en UserStories.

- Fuerte comunicación con el cliente.
- El código fuente pertenece a todos.
- Programación por parejas.
- Test como base de la funcionalidad.
- Solo el mínimo de organización.
- Pobre en cuanto a documentación(27).

FDD.

FDD es un proceso diseñado por Peter Coad, Erich Lefebvre y Jeff De Luca y se podría considerar a medio camino entre RUP (Rational Unified Process) y XP, aunque al seguir siendo un proceso ligero es más similar a este último.

FDD esta pensado para proyectos con tiempo de desarrollo relativamente cortos (menos de un año). Se basa en un proceso iterativo con iteraciones cortas (~2 semanas) que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorizar.

Puntos claves:

- Ligero.
- A medio camino entre el desarrollo y la organización.
- Existe un jerarquía dentro del equipo.
- El código fuente tiene propietario.
- Los equipos varían en función de la funcionalidad a implementar.
- El conocimiento de la aplicación se reparte a través de trabajo en equipo y revisiones.
- Documentación aceptable. (27)

RUP: Rational Unified Process

Finalmente la metodología RUP, divide en 4 fases el desarrollo del software: Inicio, Elaboración, Construcción, Transmisión. Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Las características principales del proceso son:

- Guiado por casos de uso.
- Centrado en la arquitectura.
- Iterativo e Incremental.

Puntos claves de RUP:

- Dividido en cuatro fases
- LaS faseS se dividen en iteraciones
- El discurrir del proyecto se define en Workflows
- Los artefactos son el objetivo de cada actividad
- Se basa en roles
- UML
- Muy organizativo
- Mucha documentación (27).

Finalmente la metodología que se utilizará para el desarrollo de este trabajo es RUP, por todas las ventajas y características que presenta y por política de trabajo en el proyecto.

1.4.2 Lenguaje representativo.

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE (Computer Assisted Software Engineering) orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación. En este curso se sigue el método propuesto por Craig Larman que se ajusta a un ciclo de vida iterativo e incremental dirigido por casos de uso.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.(28; 29)

1.4.3 Herramienta Case.

Las Herramientas CASE (Computer Aided Software Engineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Algunas herramientas CASE conocidas son el ArgoUML, Rational Rose, Visual Paradigm, Easy CASE, Xcase, CASE Studio 2, CASEWise entre otras. Dentro de las más conocidas se encuentra el Rational Rose y el Visual Paradigm.

Rational Rose.

Es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables. Además permite especificar, analizar y diseñar el sistema antes de codificarlo.

Principales características

- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.
- Generación de documentación automáticamente.
- Generación de código a partir de los Modelos.
- Ingeniería Inversa (crear modelo a partir código).

Visual Paradigm.

Visual Paradigm es una herramienta que sirve para realizar modelado UML. Esta herramienta tiene unas características graficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML. Permite especificar, analizar y diseñar el sistema, y posibilita la integración con diversos IDE`s (Integrated Development Environment) como: NetBeans(de Sun),JDeveloper(de Oracle ,Eclipse(de IBM) y JBuilder (de Borland).Además posibilita hacer ingeniería inversa para JAVA, .NET,

XMLY HIBERNATE. Posee versiones tanto para Windows como para Linux, por lo que se decidió valorando las ventajas del mismo ,y al no contar con la versión del Racional Rose para Linux utilizar el Visual Paradigm (30).

1.4.4 Lenguaje de programación.

Durante las últimas dos décadas, C y C++ han sido los lenguajes más utilizados para desarrollar software. Estos lenguajes ofrecen una gran flexibilidad pero, en cambio, la productividad no es muy alta ya que requieren mucho tiempo de desarrollo.

Si bien un aspecto importante a tener en cuenta es el tiempo de desarrollo otro aspecto mucho más importante es la portabilidad, es decir usar la misma aplicación en distintas arquitecturas o sistemas operativos sin tener que recompilar.

Java y C Sharp (C#) son lenguajes de programación que cumplen con las características antes mencionadas. Pero a pesar de ello actualmente Java supera a C # en cuanto a portabilidad, ya que Java utiliza el concepto de máquina virtual (MV) y su portabilidad esta verdaderamente probada. El código que se genera no es específico a una plataforma en particular. Un programa nativo: la MV se encarga de traducir este código para que en cualquier ordenador pueda ejecutarse. De esta manera un código generado en Java puede correr en cualquier plataforma, en donde se haya portado la MV.

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red. Además está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variados, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java (31).

1.4.5 Ambiente de desarrollo.

Dentro de los entornos de desarrollo integrado (IDE del inglés Integrated Development Environment) para el desarrollo de aplicaciones usando como lenguaje de programación Java se pueden encontrar NetBeans, JBuilder y Eclipse.

El NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Presenta como inconveniente que la versión actual todavía no permite la integración del entorno de desarrollo con un control de versiones Subversión.

JBuilder es otro entorno de desarrollo integrado para el lenguaje de programación Java desarrollado por Borland. Posee varias ediciones, la Enterprise, para aplicaciones J2EE, Web Services y struts. La Developer, para el desarrollo completo de aplicaciones Java, y la Foundation, con capacidades básicas para iniciarse en Java. Pero no es multiplataforma, solo puede ser ejecutado sobre el sistema operativo Windows.

La plataforma Eclipse, combinada con el JDT (Java Development Tooling), permite disponer de un IDE para Java de excelente calidad. Como IDE de Java, Eclipse posee un editor muy visual con sintaxis coloreada, ofrece compilación incremental de código, un potente depurador (que permite establecer puntos de interrupción, modificar e inspeccionar valores de variables, etc. e incluso depurar código que resida en una máquina remota), un navegador de clases, un gestor de archivos y proyectos, pero no se limita sólo a esto. La versión estándar de Eclipse proporciona también una biblioteca de refactorización de código y una lista de tareas, soporta la integración con JUnit, y suministra un front-end gráfico para Ant, la conocida herramienta de código abierto que forma parte del proyecto Jakarta de Apache.

También incluye una herramienta para completar código: el asistente de contenido, encargado de mostrar que los métodos y atributos de las clases con las que se está trabajando, ya formen parte de las APIs de Java o de cualquier otra clase en el build path, aunque estén en ficheros JAR.

Otra característica de Eclipse, muy eficiente para reducir los tiempos de depuración y pruebas, es la compilación incremental automática del código(32). Razones por las cueles se seleccionó este último IDE para el desarrollo de la herramienta.

1.5 Conclusiones

- Se desarrollará un módulo de predicción de relación estructura-actividad anticancerígena de compuestos orgánicos a partir de descriptores topológicos e híbridos.
- Se utilizará como técnica de Inteligencia Artificial la Lógica Difusa ya que se esta en presencia de información de entrada vaga, ambigua, imprecisa. Específicamente el modelo de Mamdani, ya que se ajusta a las características de nuestros datos. Se utiliza una función de pertenencia de tipo trapezoidal, se generan las reglas a partir del algoritmo Nefclass y se utiliza el método del Centro de Gravedad para defusificar.
- La aplicación se desarrollará en Java pues se necesita que la misma sea multiplataforma, se utilizará como metodología de desarrollo RUP, el Visual Paradigm como herramienta CASE y Eclipse como IDE.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En este Capítulo se describirá el modelo de dominio perteneciente a este módulo de predicción, se describen también los requisitos funcionales y no funcionales, se muestran los actores del sistema y las acciones que realizan, el diagrama de casos de uso del sistema y la descripción detallada de cada uno de estos casos de uso del sistema.

2.1 Descripción del dominio.

Al no tener un negocio bien definido, y para poder comprender mejor este sistema, se necesita la elaboración de un diagrama de clases del dominio. El mismo tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema. Este además captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el mismo.

Para este caso la idea fundamental es que el especialista que desee obtener una predicción, carga las moléculas que desea analizar y solicita hacer la predicción. Para ello, el trabajador debe haber generado anteriormente un modelo que permita al especialista realizar la predicción utilizando los datos que arroja la base de datos de descriptores. Para un mejor entendimiento se hace la propuesta del diagrama de clases del dominio, ver figura #2.

2.2 Glosario de términos del dominio.

Especialista Químico: Cliente que utilizará el sistema.
Trabajador: Persona que se encarga de generar los nuevos modelos.
Modelo: Modelo difuso que se crea a partir del las reglas difusas generadas.
Predicción: Acción que el sistema realiza para emitir un resultado.
Descriptor : Número que caracteriza estructuralmente la molécula.
Actividad Biológica: Es la actividad biológica de una molécula.
Índices topológicos: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad del grafo molecular desprovisto de hidrógeno.
Índices híbridos: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o

de conectividad en el grafo completo donde los vértices de esa matriz de conectividad se ponderan con el valor de una propiedad químico-física del tipo de átomo que separan.

Molécula: La partícula más pequeña de una sustancia, que mantiene las propiedades químicas específicas de esa sustancia.

Diagrama de Clases del Dominio

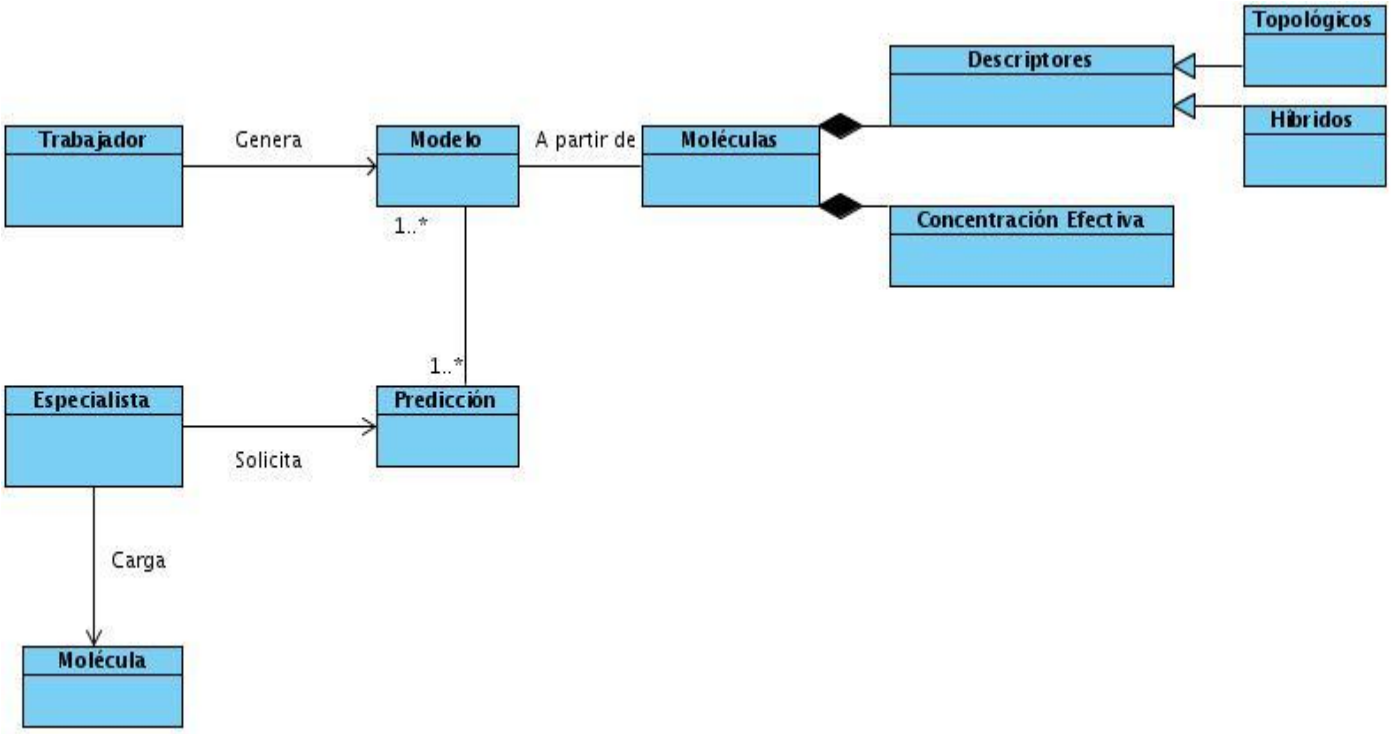


Figura #2 Representación gráfica del diagrama de clases del dominio.

2.3 Especificación de los requisitos de software.

En este epígrafe se hará la especificación de las funcionalidades que debe tener el sistema y las características que debe cumplir para que pueda ser usado.

Requisitos Funcionales.

R.1) Cargar ficheros de moléculas.

R.2) Realizar la predicción.

R.3) Mostrar predicción.

R.4) Crear modelo difuso.

R.5) Guardar modelo difuso.

R.6) Cargar información referente a las variables de independientes (cálculo de los descriptores) y dependientes (concentración efectiva).

R.7) Cargar modelo difuso.

Requisitos No Funcionales.

Requerimiento de software:

Se debe disponer de Linux, Windows 95 o superior para la instalación de la aplicación, garantizando una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas. Se necesita tener instalado el Java Rutine Enviroment (JRE) versión 1.5 o superior.

Requerimiento hardware:

Para el desarrollo y utilización del proyecto se requieren computadoras con los siguientes requisitos:

- Procesador Pentium 3 o superior.
- 256 MB de RAM o más.
- 50 MB de capacidad de disco duro

Restricciones en el diseño y la implementación.

Se debe utilizar Java como lenguaje de programación.

Requerimiento de Seguridad:

El especialista tiene control total a la aplicación sin restricción alguna. Mientras que el trabajador necesita ingresar usuario y contraseña para poder acceder a la base de datos, ya que se dispone de un mecanismo de seguridad basado en el modelo de Autenticación, Autorización y Auditoria (AAA).

Requerimiento de Usabilidad:

El sistema podrá ser usado por cualquier tipo de persona que posea conocimientos básicos en el manejo de la PC, solo se necesita que posea conocimientos de química de manera que pueda entender los resultados mostrados por la aplicación.

Requerimiento de Soporte:

- Mantenimiento: El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.
- Instalación: La instalación del sistema debe caracterizarse por su facilidad, claridad y sencillez.

Requerimiento apariencia o interfaz externa:

La aplicación debe estar diseñada con una interfaz amigable, de forma tal que el usuario navegue sin dificultad alguna, ajustándose a los estándares establecidos para el desarrollo de un buen diseño.

2.4 Definición de los casos de uso del sistema.

Este epígrafe proporciona la definición de los actores que intervienen o interactúan con el sistema. Un actor es una entidad externa del sistema que participa o interactúa con cada caso de uso. Por lo general estimula el sistema con eventos de entradas o recibe algo de él. O sea, es un rol de un usuario, que puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema.

Actores	**Justificación**
Especialista	Es el encargado de solicitar que se realice la predicción.
Trabajador	Es el encargado de actualizar el sistema creando nuevos modelos difusos.

Tabla #1 Justificación de los actores del sistema.

2.4.1 Diagrama de casos de uso del sistema.

Un diagrama de casos de uso del sistema contiene actores, casos de uso del sistema y las relaciones existentes entre los mismo. Este diagrama representa de forma gráfica como será concebido el sistema para una mejor comprensión. Refleja como interactúan los usuarios con el sistema. En la figura #3 puede observarse el diagrama casos de uso correspondiente a este sistema.

Los casos de uso definidos son:

- Realizar Predicción.
- Crear Modelo Difuso.

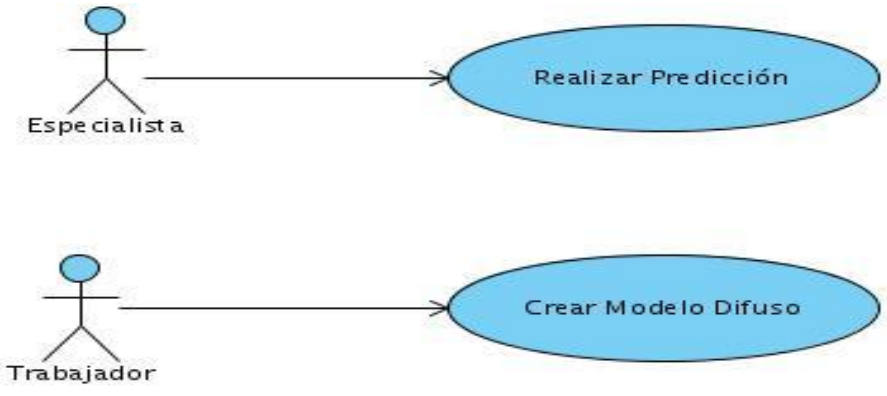


Figura #3 Representación gráfica del diagrama casos de uso del sistema.

2.4.2 Descripción de los casos de uso del sistema.

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del Diagrama de casos de uso, también se lleva a cabo la descripción textual de cada caso de uso donde se especifican todas las acciones necesarias para la realización del mismo. Con la posterior descripción de los casos de uso del sistema, se obtendrá claramente la idea de cómo se realizarán las operaciones, cuándo y quienes intervienen en ellas.

Caso de Uso:	Realizar Predicción
Actores:	Especialista Químico
Resumen:	El caso de uso inicia cuando el Especialista Químico solicita Realizar Predicción. Es aquí donde con las Reglas Borrosas(RB) y los conjuntos borrosos(CB) se realiza el procesos de inferencia obteniendo la predicción y finalizando así el caso de uso.
Referencia:	R.1, R.2, R.3,R.7
CU asociados:	**-**
Precondiciones:	Que existan el modelo difuso.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El especialista gestiona cargar los ficheros de moléculas y del modelo difuso.	1.1. El sistema carga los ficheros de las moléculas y del modelo difuso.
2. El especialista solicita Obtener Predicción.	2.1 El sistema, haciendo uso del modelo generado (Ver caso de uso Crear Modelo Difuso), realiza la predicción. 2.2 El sistema muestra la predicción.
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
Poscondiciones:	
Prioridad:	Crítico

Tabla #2 Descripción del caso de uso Realizar Predicción.

Caso de Uso:	Crear Modelo Difuso
Actores:	Trabajador
Resumen:	El caso de uso inicia cuando el Trabajador solicita Crear Modelo. Es entonces cuando éste decide trabajar directamente de la Base de Datos o subir un fichero. Luego se realiza el entrenamiento de los datos y se le permite guardar el modelo en un fichero ,finalizando asi el caso de uso.
Referencia:	R.4,R.5,R.6
CU asociados:	**-**
Precondiciones:	Que se hayan realizado los cálculos de los descriptores a cada molécula correspondiente al ensayo en la base de datos. Que haya conexión con la base de datos. Que exista el fichero con el formato necesario estructurado correctamente con los datos a analizar.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Solicitar la conexión a la base de datos.	1.1. Se conecta a la base de datos. 1.2. Carga los cálculos de los descriptores para cada compuesto de ese ensayo. 1.3. Con la información obtenida entonces crea el modelo, generando las reglas difusas y formando los conjuntos borrosos.

10 .Solicita guardar el modelo	2.1 Guarda en un fichero tanto la base de reglas como los conjuntos difusos, los que constituyen el modelo que será utilizado para realizar la predicción.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. Solicita cargar fichero correspondiente con los datos a analizar.	1.1 Carga el fichero correspondiente. 1.2 Lee el fichero con los datos de entrada.
Poscondiciones:	Que quede generado el modelo.
Prioridad:	Crítico

Tabla #3 Descripción del caso de uso Crear Modelo Difuso.

2.5 Conclusiones

En el capítulo resultó concluida la fase de descripción del negocio, realizando una modelación del dominio, por no poseer un negocio claramente definido. Se definieron los requisitos funcionales y no funcionales, diagrama de casos de uso del sistema y la descripción textual de los casos de uso del sistema. Todo esto permitió ganar en claridad en cuanto a la concepción del sistema a construir y se sentaron las bases para las restantes fases del proceso de análisis, diseño e implementación.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

En el capítulo se presentan los diagramas de clases del análisis y del diseño, dando respuesta a la solución que se propone. Se describen además los principios de diseño que se tendrán durante el desarrollo de la aplicació y se define también la arquitectura utilizada. Además se muestra el diagrama de despliegue con los elementos de configuración del procesamiento y las conexiones entre ellos.

3.1 Descripción de la arquitectura.

El estilo arquitectónico que se utilizó fue Arquitectura en 3 capas. Las motivaciones para aplicar este estilo arquitectónico estuvieron basadas fundamentalmente en el acceso a los datos y la reusabilidad ya que si en un futuro para un nuevo ensayo hay que generar nuevos modelos, el impacto se concentra en la capa de acceso a datos pero no en las otras dos.

Principales características:

- Capa de presentación: Es la que ve el usuario (conocida también como capa de usuario), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.
- Capa de negocio: Es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.
- Capa de datos: Es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

3.2 Diagrama de clases del análisis.

En este epígrafe se modela el diagrama de clases del análisis, el cual es un diagrama de clases que da una primera aproximación al diseño. Donde se muestran las clases en función de sus responsabilidades, dígase Clases Interfaz, Clases Controladoras y Clases Entidades y la relación entre ellas. Para una mejor compresión del mismo pueden verse las figuras #4 y 5, correspondientes a los casos de uso Realizar Predicción y Crear Modelo Difuso respectivamente.

Figura #4.Representación gráfica del diagrama de clases del análisis correspondiente al caso de uso Realizar Predicción.

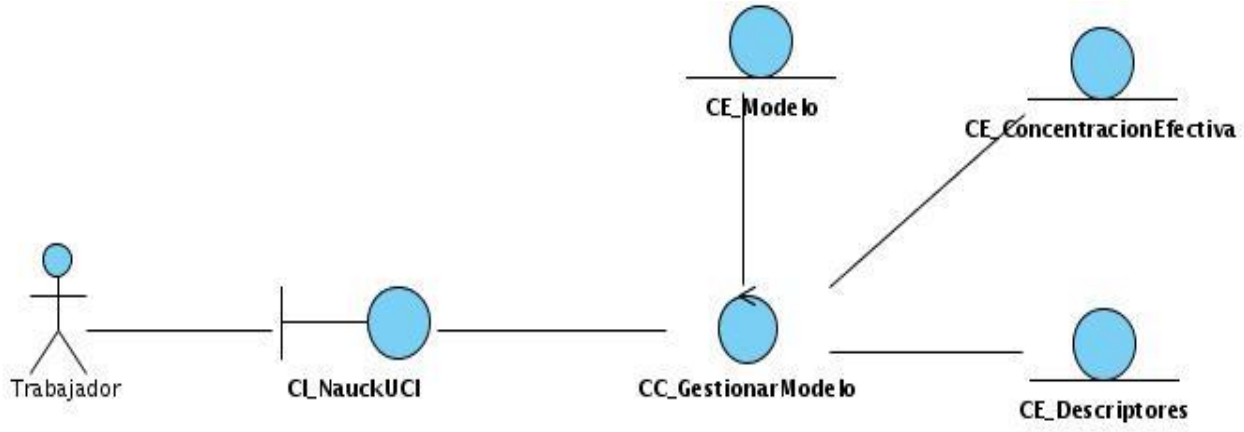


Figura #5.Representación gráfica del diagrama de clases análisis correspondiente al caso de uso Crear Modelo Difuso

3.3 Diagrama de clases del diseño.

En el presente epígrafe se presenta el diagrama de clases del diseño, el cual es un modelo de objeto que describe la realización física de los casos de uso, que en este caso son dos: Realizar Predicción y Crear Modelo. Este diagrama muestra las especificaciones para las clases de una aplicación. Incluye Clases, asociaciones y atributos así como los métodos, especificados en el lenguaje de programación con el que se realizará la implementación y las relaciones y dependencias entre las clases. Estos diagramas se pueden encontrar en las figuras # 6 y 7, correspondientes a los casos de uso Realizar Predicción y Crear Modelo Difuso respectivamente.

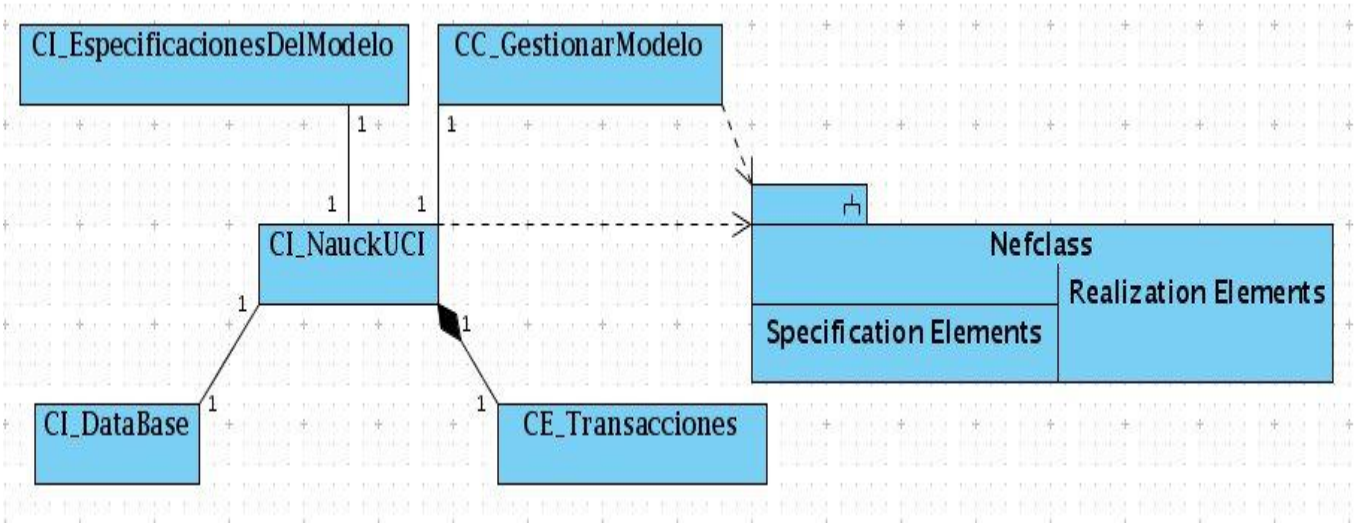


Figura #6.Representación gráfica del diagrama de clases del diseño correspondiente al caso de uso Realizar Predicción.

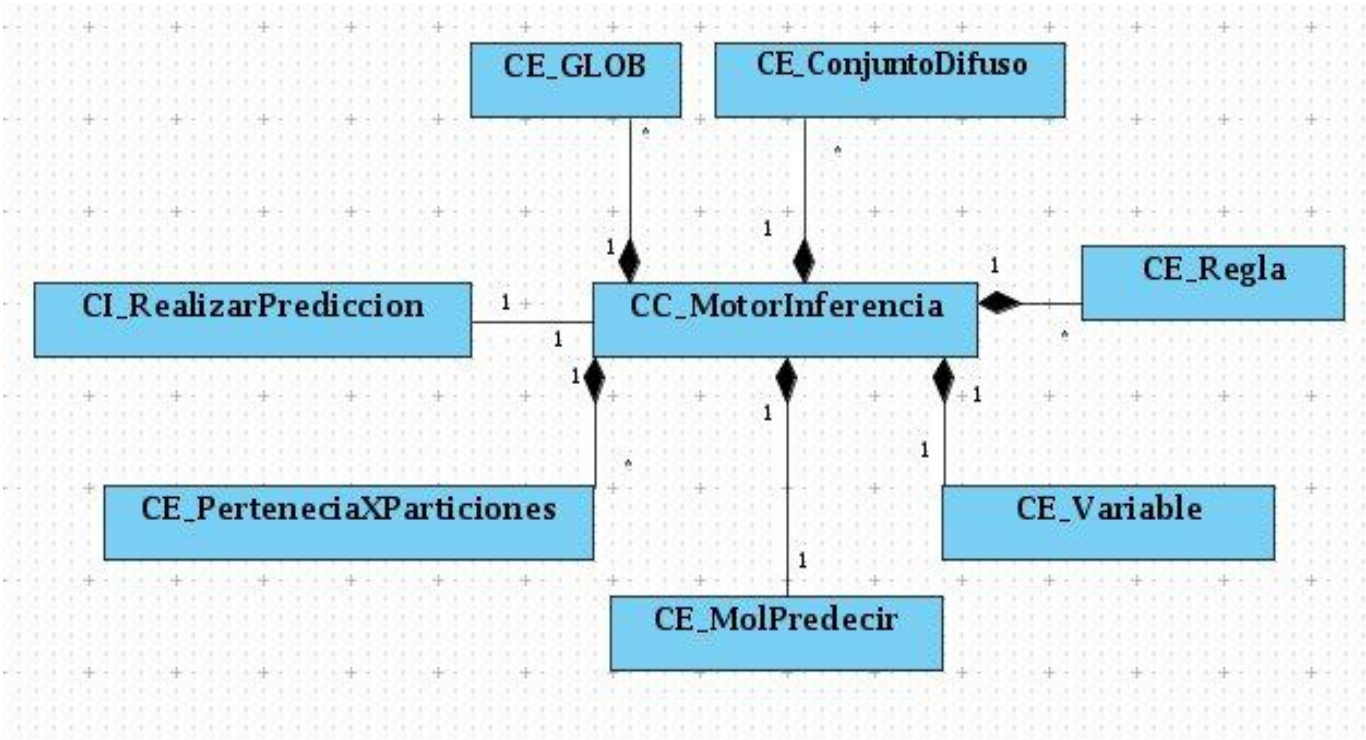


Figura #7.Representación gráfica del diagrama de clases diseño correspondiente al caso de uso Crear Modelo Difuso.

3.3 Principios de Diseño.

A continuación se describen los principios de diseño que se tuvieron en cuenta para el desarrollo de la aplicación.

Estándares en la interfaz de la aplicación.

La interfaz diseñada para el sistema está basada en el estándar de ventanas y el diseño de la aplicación es conservador, adecuado para el tipo de usuario. Logrando la fácil comprensión del mismo en el lenguaje utilizado para las opciones que brindan. Se cuenta con barras menús y botones donde se sitúan las opciones de trabajo.(Anexo #3)

Tratamiento de errores.

Los errores se le muestran al usuario de una forma clara y lo más descriptivos posibles. De forma tal que alerte de posibles riesgos al realizar determinadas operaciones.

3.4 Descripción de las clases.

Ver Anexo # 4

3.5 Diagrama de despliegue.

El Modelo de Despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre ellos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. Para este caso se cuenta con dos nodos :

- PC.
- Servidor de base de datos.

A continuación en la figura # 8 se muestra la organización y la distribución de los mismos.

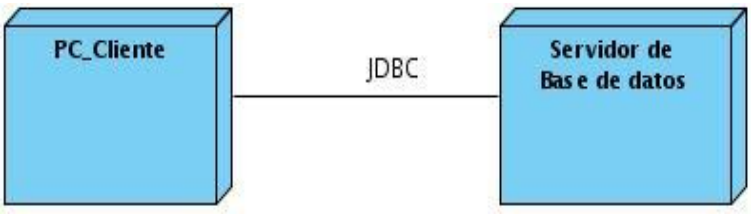


Figura #8.Representación gráfica del diagrama de despliegue.

3.6 Conclusiones.

Como resultado del estudio realizado en este capítulo, correspondiente a la etapa de análisis y diseño del sistema, se modelaron los diagramas de clases tanto del análisis como del diseño para cada uno de los casos de uso del sistema. Se establecieron las pautas para el diseño. Definiendo atributos y métodos necesarios para realizar la implementación especificada en el lenguaje Java y finalmente se mostró el diagrama de despliegue.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se muestran los principales artefactos del flujo de trabajo de implementación y de prueba. Teniendo como base el resultado del diseño se implementa el sistema en términos de componentes. Y se realizan las pruebas y validaciones necesarias para demostrar que la aplicación posee la calidad requerida y cumple con los objetivos propuestos.

4.1 Diagrama de componentes.

A continuación se muestra el diagrama de componentes correspondiente a la aplicación. En este diagrama se pueden mostrar los componentes como ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares, entre otros. En el flujo de trabajo de diseño se propone crear un plano del modelo de implementación. El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

Un diagrama de componentes muestra además las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. En la figura # 9 se muestra el diagrama de componentes de la herramienta desarrollada.

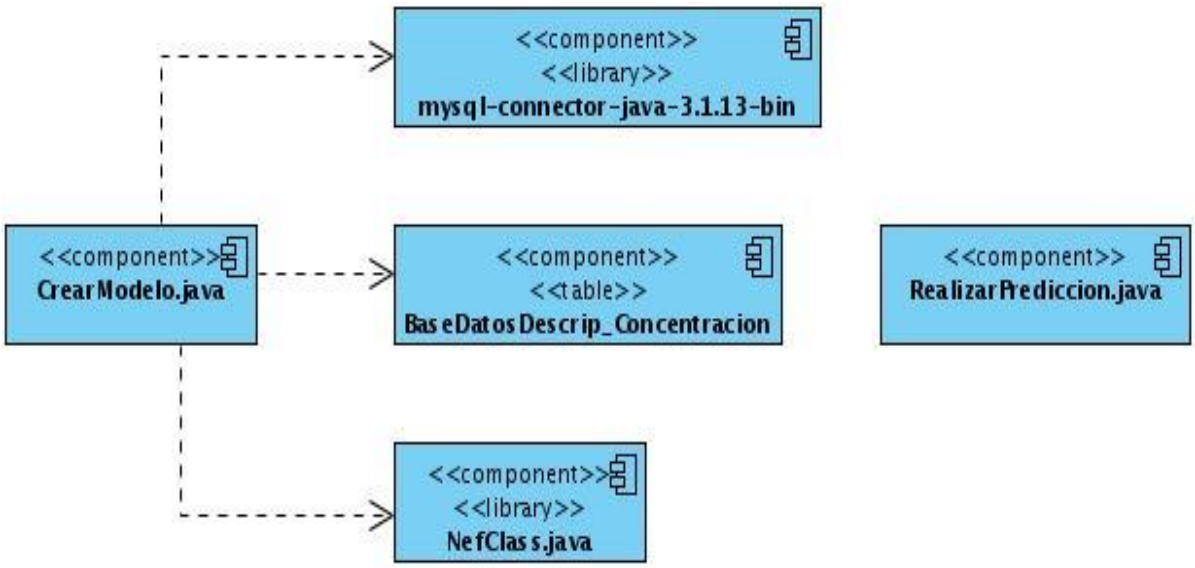


Figura #9.Representación gráfica del diagrama de componentes.

4.2 Pruebas.

Las pruebas son una actividad en la cual un sistema o componente se ejecuta bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y se hace una evaluación de algún aspecto del sistema o componente. Las pruebas no pueden asegurar la ausencia de defectos; solo pueden demostrar que existen defectos en el software.

Las pruebas que se realizaron en el presente trabajo de diploma están referidas a validar la calidad del modelo que se crea por la aplicación y ver la capacidad de la lógica difusa para clasificar los compuestos. Se utilizaron dos muestras, la primera de tamaño 1600 perteneciente a la base de datos PubMed (del National Center for Biotchnology Information), la cual es la base de datos utilizada por el proyecto para realizar las investigaciones; y la segunda muestra, de tamaño 150, perteneciente a la base de datos IRIS(de U.S. Environmental Protection Agency (EPA) and its Office of Research and Development, National Center for Environmental Assessment). IRIS es una base de datos de los efectos

en salud humana que pueden resultar de la exposición a varias sustancias encontradas en el medio ambiente.

4.2.1 Pruebas muestra 1.

El ensayo utilizado fue :"NCI Yeast Anticancer Drug Screen. Data for the rad50 strain" tomado de la base de datos PubMed. Este ensayo cuenta con 6112 moléculas activas y 51532 moléculas inactivas. Para realizar las pruebas se tomó una muestra de 2000 compuestos orgánicos entre activos e inactivos, manteniendo la misma proporción que existe en la base de datos, el cual revela que por cada molécula activa hay aproximadamente 8 moléculas inactivas. La muestra se redujo debido a que se necesitan máquinas con mayor memoria RAM para procesar esa gran cantidad de datos. Esa muestra se dividió en muestras de entrenamiento (80%) para generar el modelo, donde 170 eran compuestos activos y 1430 inactivos y otra de muestra de prueba (20%) para evaluar el modelo. El modelo generado reportó la siguiente información:

Clasificación de los datos:

Análisis por clases:

	0	1	n.c	Suma
0	52 (30.59%)	89 (52.35%)	29 (17.06%)	170 (100% de activos)
1	39 (2.72%)	1213 (84.83%)	178 (12.45%)	1430 (100% de inactivos)
suma	91 (5.68%)	1302 (81.38%)	207 (12.94%)	1600 (100% total)

Tabla# 4: Reporte de clasificación de la aplicación. Muestra1

Correctos: 1265 (79.06%), Incorrectos: 335 (20.94%)

0: Activas

1: Inactivas

n.c.: No clasificados

Puede observarse que los resultados obtenidos al clasificar los compuestos activos no son buenos, razón por la cual se decidió realizar pruebas sobre el software Weka(Waikato Environment for Knowledge

Analysis) utilizando diferentes métodos. Este software es muy utilizado y realmente reconocido entre los de su tipo.

Weka, es un conjunto de librerías Java para la extracción de conocimientos desde bases de datos. Este es un software que ha sido desarrollado bajo licencia GPL lo cual ha impulsado que sea una de las suites más utilizadas en el área en los últimos años (33).

Se estableció para efectuar la validación del modelo aprendido Use training set, ya que con esta opción Weka entrenará el método con todos los datos disponibles y a posteriori realiza la evaluación sobre los mismos datos.

Los métodos para clasificación que se utilizaron fueron los métodos matemáticos: Redes neuronales, específicamente los perceptrones multicapa y las redes de funciones de base radial, los que arrojaron los siguientes resultados:

Scheme: weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
= = =Confusion Matrix= = =

a	b	Classified as
26	144	a=activa
12	1418	b=inactiva

Tabla# 5: Resultados de clasificación utilizando perceptrones multicapa. Muestra 1

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
= = =Confusion Matrix= = =

a	b	Classified as
0	170	a=activa
0	1430	b=inactiva

Tabla#6 : Resultados de clasificación utilizando redes de funciones de base radial. Muestra1

Por otro lado se utilizó un algoritmo basado en árboles de decisión el C4.5 el cual aparece el Weka como J48 y reportó los siguientes resultados.

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

=== Confusion Matrix ===

a	b	Classified as
29	141	a=activa
1	1429	b=inactiva

Tabla# 7: Resultados de clasificación utilizando C4.5.Muestra1

Luego de realizar las pruebas sobre el Weka y hacer comparaciones contra los resultados que reveló la aplicación, se concluyó que el problema que existe al clasificar los compuestos activos como inactivos no esta dado por problemas de programación o por utilizar procedimientos erróneos, sino que está dado porque el espacio de parámetros de los descriptores no cubre los compuestos activos, por lo que habría que incorporar nuevos descriptores con otro contendido de información estructural.

4.2.2 Pruebas muestra 2.

Otra de las pruebas realizadas sobre la aplicación fueron utilizando otra muestra perteneciente de la base de datos IRIS.

Para la prueba se utilizaron 4 variables de entrada y 3 de salida, con un tamaño de muestra de 150.

Al generar el modelo difuso se obtuvo la siguiente información.

	0	1	2	n.c	Suma
0	50(100%)	0(0%)	0(0%)	0(0%)	50 (100% total class 1)
1	0(0%)	48(96%)	2(4%)	0(0%)	50 (100% total class 2)
2	0(0%)	3(6%)	45(90%)	2(4%)	50 (100% total class 3)
Suma	50(33.33%)	51(34%)	47(31.33%)	2 (1.33%)	150(100% total)

Tabla# 8: Resultados de clasificación utilizando la aplicación con la muestra de la base de datos IRIS.

0: Class 1

1: Class 2

2: Class 3

n.c.: No clasificados

Se observó que al generar el modelo difuso con estos datos la clasificación es buena, obteniendo:

Correctos: 143 (95.33%), Incorrectos: 7 (4.67%).

También se probó en el weka la muestra anteriormente analizada con los mismos algoritmos, y dio el siguiente resultado:

Scheme: weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

=== Confusion Matrix ===

a	b	c	Classified as
50	0	0	a = Iris-setosa
0	49	1	b = Iris-versicolor
0	1	49	c = Iris-virginica

Tabla#9: Resultados de clasificación utilizando perceptrones multicapa. Muestra 2

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1

= = =Confusion Matrix= = =

a	b	c	Classified as
50	0	0	a = Iris-setosa
0	47	3	b = Iris-versicolor
0	1	49	c = Iris-virginica

Tabla#10 : Resultados de clasificación utilizando redes de funciones de base radial. Muestra 2

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2

=== Confusion Matrix ===

a	b	c	Classified as
50	0	0	a = Iris-setosa
0	47	3	b = Iris-versicolor
0	1	49	c = Iris-virginica

Tabla# 11: Resultados de clasificación utilizando C4.5.Muestra 2

Confirmándose con estos buenos resultados de la muestra 2 que el problema presentado al clasificar mal los compuestos activos de la muestra 1 está dado por la naturaleza de los datos de entrada.

4.2.3 Pruebas de predicción.

Ahora se mostrarán las pruebas correspondientes a la predicción realizada con la muestra 1. Estos resultados van a estar en correspondencia con el modelo utilizado, usando la lógica difusa como técnica para la clasificación y 72 descriptores de entrada, se evaluaron 401 compuestos correspondientes a la muestra de prueba, de los cuales 43 eran activos y 358 inactivos.

	Clasificadas como Activas	Clasificadas como Inactivas	Total
Moléculas Activas	2(4,65%)	41(95,35%)	43(100% de activos)
Moléculas Inactivas	5(1,4%)	353(98,6%)	358(100% de inactivos)
sum	7	394	401

Tabla#12: Resultados de la clasificación utilizando la aplicación desarrollada.

Puede observarse que se logró clasificar aceptablemente los compustos inactivos con más del 90% de acierto, no siendo así con los compuestos activos.

4.3 Conclusiones.

Como resultado de este capítulo, se modelaron los diagramas de componentes, se realizaron las pruebas correspondientes a la validación del modelo, y se realizaron comparaciones entre los modelos generados por la aplicación desarrollada y algoritmos implementados en el software Weka.

CONCLUSIONES GENERALES

- Se analizó, diseñó e implementó un sistema de predicción de actividad biológica basado en lógica difusa empleando descriptores topologicos e híbridos como modo de descripción de la estructura química.
- Se desarrolló un modelo de predicción sobre una muestra de entrenamiento de 1600 compuestos, porbados con 401 compuestos de la muestra de prueba, todos pertenecientes al ensayo :"NCI Yeast Anticancer Drug Screen. Data for the rad50 strain", de la base de datos PubMed. Se demostró la capacidad del modelo para clasificar correctamente los compuestos inactivos (85% del total de inactivos de la muestra) y los activos (30.6% del total de activos de la muestra) , lo cual se atribuye a limitaciones en la capacidad descriptora de las variables utilizadas.
- Se obtuvieron resultados coincidentes al evaluar la aplicación con datos externos y compararlos con los realizados utilizando el NefClass y el Weka, y al evaluar el modelo obtenido en el presente trabajo con el Weka se comprobó tanto el buen funcionamiento de la aplicación como la calidad del modelo desarrollado.
- Se implementó una aplicación como plug-in para su incorporación al visualizador del sistema para la predicción de actividad biológica de compuestos orgánicos utilizando como lenguaje de programación Java.

RECOMENDACIONES

Se recomienda para la siguiente etapa:

- Incorporar al estudio nuevos descriptores tanto topológicos como topográficos.
- Incorporar nuevos ensayos.
- Realizar un análisis estadístico de varianza y desviación típica de las variables independientes (descriptores) con la finalidad de utilizar aquellas que realmente aporten información relevante.
- Utilizar la aplicación en medios de cómputo más potentes empleando mayores volúmenes de datos, para verificar que se creen mejores modelos difusos.

REFERENCIAS BIBLIOGRAFICAS

1. J.C.ESCALONA, R. C., J.A.PADRÓN. Folleto para la docencia. En Introducción al diseño de fármacos.

2. Codessa. [Consultado el: Noviembre 2006]. Disponible en: http://www.semichem.com/codessa.html

3. Dragon. [Consultado el: Noviembre 2006]. Disponible en: http://www.talete.mi.it/products/dragon_all_about.htm.

4. Molconn-Z. [Consultado el: Noviembre 2006]. Disponible en: http://www.edusoft-lc.com/molconn/.

5. HERNÁNDEZ, N. R. F. Sistema "GRATO (GRAph-TOol)" para la Visualización Molecular y el Cálculo de Descriptores. Trabajo de Diploma, Junio,2006.

6. HERIBERTO CASTAÑETA , E. A. C. Aplicación de los métodos QSAR/QSPR en fenómenos de adsorción de sustancias químicas sobre materiales sólidos. publicado el: Diciembre 2006 de última actualización: Diciembre 2006. Disponible en: http://www.monografias.com/trabajos41/metodos-qsar-qspr/metodos-qsar-qspr.shtml.

7. ALAIN ZARRAGOITIA GONZÁLEZ, Y. M. A. G., MAYRA REYES MORENO, HAROLD CURIEL HERNÁNDEZ, EDDY CASTELLANOS GIL, ULISES J. JÁUREGUI HAZA, JOSE A. RUIZ GARCÍA. Sustitución del diclorometano por etanol en una etapa del proceso de síntesis de la oximetolona. publicado el: Enero 2006 de última actualización: Enero 2006. Disponible en: http://bvs.sld.cu/revistas/far/vol36_s_02/G%20Publicaciones%20QF-Sintesis.pdf.

8. PORRAGAS, G. E. Modelos QSPR/QSAR/QSTR basados en sistemas neuronales cognitivos Junio 2002.

9. History of Quantitative Structure-Activity Relationships Burger's Medicinal Chemistry and Drug Discovery, 2003, vol. 1, nº [Consultado el: Febrero,2007]. Disponible en: http://media.wiley.com/product_data/excerpt/03/04712709/0471270903.pdf. ISSN 0-471-27090-3

10. CASTAÑÓN, H. G. D. T. Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador. 2004.

11. JUAN LUCAS DOMÍNGUEZ RUBIO, M. J. C. B., WLADIMIRO DÍAZ VILLANUEVA. Discriminación y predicción de propiedades de fármacos mediante redes neuronales. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003, vol. 18, nº [Consultado el: Noviembre,2006]. Disponible en: http://wotan.liu.edu/docis/dbl/iariia/2003_18_7_DYPDPD.htm.

12. Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. Journal of chemical information and modeling, Abril 2004 nº [Consultado el: Enero 2007]. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i04/abs/ci049965i.html.

13. PÉREZ, P. Y. P. Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing. Tesis Doctoral, 2005.

14. Fuzzy structure – activity relationships Septiembre,2002 2002, nº [Consultado el: Febrero,2007].

15. Apex-3D expert system for drug design. publicado el: abril,2007 de última actualización: abril,2007. Disponible en: http://www.netsci.org/Science/Compchem/feature09.html

16. MCASE. Disponible en: http://www.multicase.com/products/prod01.htm.

17. Oncologic,ECOSAR Y AQUATOX. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://translate.google.com/translate?hl=es&sl=en&u=http://epa.gov/oppt/ar/20052006/cross_cutting/ces.htm&sa=X&oi=translate&resnum=1&ct=result&prev=/search%3Fq%3DOncologic%252BEPA%2527s%2BOffice%26hl%3Des%26sa%3DG.

18. Other Software. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://lem.ch.unito.it/cl/category.php?catcode=7.

19. GÓMEZ, J. G. Conjuntos y Sistemas Difusos(Lógica Difusa y Aplicaciones). publicado el: Diciembre,2006 de última actualización: Diciembre,2006. Disponible en: http://www.lcc.uma.es/~ppgg/FSS/.

20. Inteligencia Artificial En Sistemas basados en el conocimiento 2006.

21. QUISPE, J. C. C. Control neuro_difuso aplicado a una grúa Torre. publicado el: Mayo,2007 de última actualización: Mayo,2007. Capítulo 2 p.

22. SARMIENTO, J. A. A. Aprendizaje de particiones difusas para razonamiento inductivo Tesis doctoral, 2006.

23. Metodología de Desarrollo de Software (MDS),ublicado el: Enero,2007 de 2005, última actualización: Enero,2007. Disponible en: http://reynox.com/sistemas/metodologia.php.

24. VIDAL, T. A. Introducción a la Lógica Difusa (Fuzzy Logic). . 2005,

25. PEDRO YOBANIS PIÑERO PÉREZ, L. A. G., MARIA M. GARCÍA LORENZO, YAILÉ CABALLERO MOTA, RAFAEL BELLO PÉREZ. Aprendizaje de reglas borrosas a partir de Casos para la clasificación automatizada.Bioinformática y otras aplicaciones de la ciencia. publicado el: Noviembre.2006 de última actualización: Noviembre.2006.

26. KRUSE, P. D. R.; GRAUEL, P. D. A., et al. Data Analysis with Neuro-Fuzzy Methods. Doctoral, 2000.

27. MOLPECERES, A. Procesos de desarrollo: RUP, XP y FDD. Disponible en: http://www.javahispano.org/articles.article.action?id=76.

28. UML. publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.clikear.com/manuales/uml/introduccion.asp

29. Modelado de Sistemas con UML. Disponible en: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/.

30. Visual Paradigm for UML. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://www.programacion.net/noticia/1363/.

31. Que es Java .Características del lenguaje Java. publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html

32. ABIÁN., M. Á. El archipiélago ECLIPSE (PARTE 2 DE 4) publicado el: Febrero,2007 de 2003, última actualización: Febrero,2007. Disponible en: http://www.javahispano.org/articles.article.action?id=81

33. Práctica de Minería de Datos.Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software. publicado el: Junio,2007 de 2006, última actualización: Junio,2007.

BIBLIOGRAFIA

1. Codessa. [Consultado el: Noviembre 2006]. Disponible en: http://www.semichem.com/codessa.html

2. Dragon. [Consultado el: Noviembre 2006]. Disponible en: http://www.talete.mi.it/products/dragon_all_about.htm.

3. Molconn-Z. [Consultado el: Noviembre 2006]. Disponible en: http://www.edusoft-lc.com/molconn/.

4. UML. publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.clikear.com/manuales/uml/introduccion.asp

5. Modelado de Sistemas con UML. Disponible en: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/.

6. Qué es Java .Características del lenguaje Java. publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html

7. Clasificación de patrones.

8. Sistemas difusos. En Sistemas basados en reglas difusas.

9. Apex-3D expert system for drug design. publicado el: abril,2007 de última actualización: abril,2007. Disponible en: http://www.netsci.org/Science/Compchem/feature09.html

10. PASS: An Expert System with Certainty Factors for Predicting Student Success publicado el: abril,2007 de última actualización: abril,2007. Disponible en:

http://www.springerlink.com/content/7gmb2xt1b8bkhr05/.

11. MCASE. Disponible en: http://www.multicase.com/products/prod01.htm.

12. Other Software. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://lem.ch.unito.it/cl/category.php?catcode=7.

13. Oncologic,ECOSAR Y AQUATOX. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://translate.google.com/translate?hl=es&sl=en&u=http://epa.gov/oppt/ar/20052006/cross_cutting/ces.htm&sa=X&oi=translate&resnum=1&ct=result&prev=/search%3Fq%3DOncologic%252BEPA%2527s%2BOffice%26hl%3Des%26sa%3DG.

14. Visual Paradigm for UML. publicado el: Mayo,2007 de última actualización: Mayo,2007. Disponible en: http://www.programacion.net/noticia/1363/.

15. Flujo de implementación.

16. FT Análisis y Diseño. Modelo de Diseño.

17. Fuzzy structure – activity relationships Septiembre,2002 2002, nº [Consultado el: Febrero,2007].

18. History of Quantitative Structure-Activity Relationships Burger's Medicinal Chemistry and Drug Discovery, 2003, vol. 1, nº [Consultado el: Febrero,2007]. Disponible en: http://media.wiley.com/product_data/excerpt/03/04712709/0471270903.pdf. ISSN 0-471-27090-3

19. Aprendizaje: preliminares publicado el: Enero 2007 de 2005, última actualización: Enero 2007. Disponible en: http://www.gsi.dit.upm.es/~gfer/ssii/preliminares.pdf.

20. Metodología de Desarrollo de Software (MDS) publicado el: Enero,2007 de 2005, última

actualización: Enero,2007. Disponible en: ttp://reynox.com/sistemas/metodologia.php.

21. Práctica obligatoria. publicado el: Febrero,2007 de 2005-2006, última actualización: Febrero,2007.

22. Inteligencia Artificial En Sistemas basados en el conocimiento. . 2006.

23. Práctica de Minería de Datos.Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software. publicado el: Junio,2007 de 2006, última actualización: Junio,2007.

24. Comparative study of QSAR/QSPR correlations using support vector machines, radial basis function neural networks, and multiple linear regression. Journal of chemical information and modeling, Abril 2004 nº [Consultado el: Enero 2007]. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i04/abs/ci049965i.html.

25. .PEDRO YOBANIS PIÑERO PÉREZ, L. A. G., MARIA M. GARCÍA LORENZO, YAILÉ CABALLERO MOTA, RAFAEL BELLO PÉREZ. Aprendizaje de reglas borrosas a partir de casos para la clasificación automatizada.Bioinformática y otras aplicaciones de la ciencia. publicado el: Noviembre.2006 de última actualización: Noviembre.2006.

26. ABIÁN., M. Á. El archipiélago ECLIPSE (PARTE 2 DE 4) publicado el: Febrero,2007 de 2003, última actualización: Febrero,2007. Disponible en: http://www.javahispano.org/articles.article.action?id=81

27. ALAIN ZARRAGOITIA GONZÁLEZ, Y. M. A. G., MAYRA REYES MORENO, HAROLD CURIEL HERNÁNDEZ, EDDY CASTELLANOS GIL, ULISES J. JÁUREGUI HAZA, JOSE A. RUIZ GARCÍA. Sustitución del diclorometano por etanol en una etapa del proceso de síntesis de la oximetolona. publicado el: Enero 2006 de última actualización: Enero 2006. Disponible en: http://bvs.sld.cu/revistas/far/vol36_s_02/G%20Publicaciones%20QF-Sintesis.pdf.

28. ALBERTO FERNÁNDEZ HILARIO, S. G., FRANCISCO HERRERA,MARÍA JOSÉ DEL JESUS. Un primer estudio sobre el uso de los Sistemas de Clasificación basados en reglas difusas en problemas de clasificación con clases no balanceadas. publicado el: Febrero,2007 de última actualización: Febrero,2007.

29. ALTAMIRANO, A. V. Comparativo de entornos de desarrollo integrados (IDE's). publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.ubicuos.com/request.php?2.

30. AURORA VIZCAINO, I. C. Una Herramienta CASE para ADOO:Rational Rose.

31. BÉJAR, J. Inteligencia Artificial (IA) .Apuntes de Aprendizaje Automático . publicado el: Enero 2007 de 2006, última actualización: Enero 2007. Disponible en: http://www.lsi.upc.es/~bejar/ia/material/teoria/aprendizaje.pdf.

32. CALVO, O. Clasificador no supervisado MAX-MIN. 2000, [Consultado el: Febrero,2007].

33. Clasificador Supervisado basado en Perceptrón Simple. 2000, [Consultado el: Febrero,2007].

34. CARVAJAL, P. H. y ARENAS, L. G. Propuesta sistema neurodifuso de clasificación para optimización del modelo de evaluación de calidad de productos de software. publicado el: Febrero,2007 de última actualización: Febrero,2007.

35. CASTAÑÓN, H. G. D. T. Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador. 2004.

36. CÉSAR VIDAL, J. M. G.-G., LUIS MARTÍ-BONMATÍ, ALFONS JUAN, MONTSERRAT ROBLES. Clasificación de estirpes histológicas de tumores de partes blandas mediante reconocimiento de patrones a partir de imágenes de RM. nº [Consultado el: Febrero,2007].

37. EYE, A. V. Comparing Tests of Multinormality - A Monte Carlo Study. 2005,

38. FERNANDO ARÁMBULA COSÍO1, J. M. F., MIGUEL ANGEL PADILLA CASTAÑEDA1, PATRICIA TATO2, SANDRA SOLANO2. Análisis automático de muestras de tejido teñidas inmunocitoquímicamente. publicado el: Febrero,2007 de 2005, última actualización: Febrero,2007.

39. GALIANO, F. B. Aprendizaje de clasificadores. publicado el: Febrero,2007 de última actualización: Febrero,2007.

40. GÓMEZ, J. G. Conjuntos y Sistemas Difusos(Lógica Difusa y Aplicaciones). publicado el: Diciembre,2006 de última actualización: Diciembre,2006. Disponible en: http://www.lcc.uma.es/~ppgg/FSS/.

41. GÓMEZ, R. S. Inteligencia en redes de comunicaciones. publicado el: Octubre,2006 de última actualización: Octubre,2006. [Consultado el: Octubre,2006].

42. HENAO, J. D. V. Prónostico de la serie de MACKEY-GLASS usando modelos de regresión no lineal . Dyna,Red de Revistas Científicas de América Latina y el Caribe, España y Portuga, Julio 2004, vol. 71, nº [Consultado el: Febrero,2007]. ISSN 0012-7353.

43. HERIBERTO CASTAÑETA , E. A. C. Aplicación de los métodos QSAR/QSPR en fenómenos de absorción de sustancias químicas sobre materiales sólidos. publicado el: Diciembre 2006 de última actualización: Diciembre 2006. Disponible en: http://www.monografias.com/trabajos41/metodos-qsar-qspr/metodos-qsar-qspr.shtml.

44. HERNÁNDEZ, N. R. F. Sistema “GRATO (GRAph-TOol)” para la Visualización Molecular y el Cálculo de Descriptores. Trabajo de Diploma, Junio,2006.

45. J.C.ESCALONA, R. C., J.A.PADRÓN. Folleto para la docencia. En Introducción al diseño de fármacos.

46. JAVIER BEJAR, U. C., MIQUEL SANCHEZ-MARRE. Aprendizaje Automático. publicado el: Enero, 2007 de última actualización: Enero, 2007. Disponible en: http://www.lsi.upc.es/~webia/gr-SBC/bib/articles/bejar/novatica/novatica.ps.gz.

47. JIMÉNEZ. M, D.-D. R., SORIGUER. R.C, PRADO. E, GARCÍA. A. Diversidad biológica del matorral de la reserva biológica de Doñana mediante imágenes hiperespectrales aeroportadas AHS . publicado el: Febrero,2007 de última actualización: Febrero,2007.

48. JUAN LUCAS DOMÍNGUEZ RUBIO, M. J. C. B., WLADIMIRO DÍAZ VILLANUEVA. Discriminación y predicción de propiedades de fármacos mediante redes neuronales. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 2003, vol. 18, nº [Consultado el: Noviembre,2006]. Disponible en: http://wotan.liu.edu/docis/dbl/iariia/2003_18_7_DYPDPD.htm.

49. KRUSE, P. D. R.; GRAUEL, P. D. A., et al. Data Analysis with Neuro-Fuzzy Methods. Doctoral, 2000.

50. MARÍA GABRIELA DÍAZ-ANTÓN, M. A. P., ANNA C. GRIMMÁN, LUIS E. MENDOZA Propuesta de una metodología de desarrollo de software educativo bajo un enfoque de calidad sistémica. publicado el: Febreo,2007 de última actualización: Febreo,2007. Disponible en: http://www.academia-interactiva.com/ise.pdf.

51. MAXWELLING. Fisher Linear Discriminant Analysis. publicado el: Febrero,2007 de última actualización: Febrero,2007.

52. MOLPECERES, A. Procesos de desarrollo: RUP, XP y FDD. Disponible en: http://www.javahispano.org/articles.article.action?id=76.

53. MYUNG, I. J. Tutorial on maximum likelihood estimation. publicado el: Febrero,2007 de 2002, última actualización: Febrero,2007.

54. ORALLO, E. H. El lenguaje unificado de modelado (UML) . publicado el: Febrero,2007 de última actualización: Febrero,2007. Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

55. PEDRO Y. PIÑERO, L. A., MARÍA M. GARCÍA, LIESNER ACEVEDO. Algoritmos genéticos en la construcción de funciones de pertenencia borrosas. . publicado el: Enero,2007 de última actualización: Enero,2007.

56. PÉREZ, J. C. Y. J. C. OCR (Optical Character Recognition). publicado el: Febrero,2007 de última actualización: Febrero,2007.

57. PÉREZ, P. Y. P. Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing. Tesis Doctoral, 2005.

58. PORRAGAS, G. E. Modelos QSPR/QSAR/QSTR basados en sistemas neuronales cognitivos . Junio 2002.

59. QUISPE, J. C. C. Control neuro_difuso aplicado a una grúa Torre. publicado el: Mayo,2007 de última actualización: Mayo,2007. Capítulo 2 p.

60. S.PURCELL. Maximum Likelihood Estimation. publicado el: Febrero,2007 de 2000, última actualización: Febrero,2007.

61. SÁNCHEZ, J. S. Aprendizaje y clasificación basados en criterios de vecindad. Métodos alternativos y análisis comparativo. . Tesis Doctoral

62. SARMIENTO, J. A. A. Aprendizaje de particiones difusas para razonamiento inductivo. Tesis doctoral, 2006.

63. SOUDAN, M. M. Outlier Detection of Multivariate Extrapolated Administrative Data in Belgian

National Accounts. En 2002

64. TEKNOMO, K. Euclidean Distance. publicado el: Febrero,2007 de 2006, última actualización: Febrero,2007. Disponible en: http://people.revoledu.com/kardi/tutorial/Similarity/EuclideanDistance.html.

65. VALENCIA, C. A. C. Evaluación de algoritmos basados en lógica difusa aplicados al preproceso y detección de bordes en imágenes digitales. publicado el: Febrero,2007 de 2004, última actualización: Febrero,2007. [Consultado el: Febrero,2007].

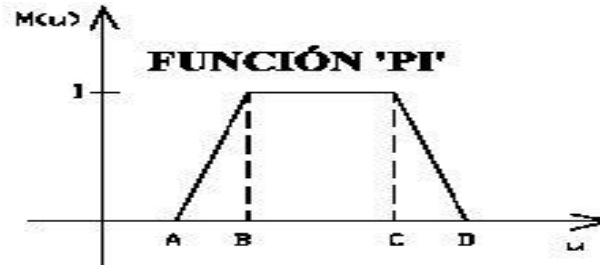
66. VIDAL, T. A. Introducción a la Lógica Difusa (Fuzzy Logic). 2005,

ANEXOS

Anexo#1

Función Pi: Funciones trapezoidales

Estas funciones pueden verse como una extensión o generalización de las funciones Gamma, L y LAMBDA. Esta función Pi se representa en el siguiente gráfico, y su nombre viene dado por su parecido con esta letra griega PI Π.



Esta función es característica de las etiquetas que significa el adjetivo que caracteriza a los elementos centrales del universo del discurso, pero que se quiere permitir que valga 1 en más de un punto (a diferencia de las funciones triangulares en las que sólo vale 1 en un punto).

Estas funciones formalmente quedarían:

0	si X <= A
M(X)= (X-A)/(B-A)	si X > A y si X < B
1	si X >= B y si X =< C
(D-X)/(D-C)	si X > C y si X < D
0	si X >=C

Anexo#2

Operaciones sobre conjuntos difusos.

Las operaciones lógicas que se pueden establecer entre conjuntos difusos son la intersección (conjunción), la unión (disyunción) y el complemento (negación), igual que las usadas en lógica bivalente o clásica. Las operaciones con conjuntos difusos darán como resultado otros conjuntos también difusos. Dado que la lógica difusa es una extensión de la bivalente o clásica, las nuevas operaciones que se explican para interceptar o unir conjuntos difusos son también aplicables a la lógica bivalente o clásica obteniendo idénticos resultados.

La teoría difusa está formulada muchas veces como conjuntos y por eso se habla muchas veces en términos de las operaciones de conjuntos (intersección, unión y el complemento), sin embargo, en todos los casos esto aquí es equivalente a la forma en la que se ha nombrado en la lógica o en la Teoría de la Incertidumbre (conjunción, disyunción y negación).

En lógica difusa hay muchas maneras de definir estas operaciones. Cualquier operación que cumpla las restricciones de una T-Norma puede ser usada para interceptar, igual que cualquier S-Norma puede ser usada para unir conjuntos difusos. Las T-Normas especifican un conjunto de condiciones que deben reunir aquellas operaciones que deseen ser usadas para interceptar conjuntos, mientras que las S-Normas hacen lo propio para las uniones.

Operadores básicos

Intersección: la intersección de dos conjuntos se obtiene aplicando la operación T-Norma que puede ser sustituida por cualquiera de las funciones posibles asociadas a ella (las que se muestran posteriormente. El proceso de intersección visto desde el punto de vista de la Teoría de Hajek es lo que ocurre en las funciones de Hajek de CONJ (conjunción) y CTR (contribución).

Unión: la unión de dos conjuntos difusos se obtiene aplicando la operación S-Norma que puede ser sustituida por cualquiera de las funciones que posibles asociadas a ella (las que se muestran posteriormente). El proceso de unión visto desde el punto de vista de la Teoría de Hajek es lo que ocurre en las funciones de Hajek de DISY (disyunción) y GLOB (global).

Negación o complemento: la negación o complemento de un conjunto difuso se obtiene aplicando esta fórmula, dado que los valores de la función de pertenencia oscilan entre 0 y 1. El proceso de negación (complemento) visto desde el punto de vista de la Teoría de Hajek es lo que ocurre en la función de Hajek de NEG (negación). Es importante notar que en la Teoría Difusa no se puede usar la función de Hajek NEG(X)= -X ya que implicaría obtener valores negativos. Por eso aquí se hace el mismo proceso de reflexión alrededor del centro que se hace en Hajek, pero tomando el 0.5 como centro en lugar del 0 como es en Hajek. De esta manera, la función difusa de NEG es NEG(X) = 1 - X. Por ejemplo, si la etiqueta "alto" vale 0 para un caso, su negación vale NEG(0) = 1 - 0=1. Esto significa que si es "alto" con 0, es lo mismo que decir que "no es alto" con 1.

T-Norma y S-Norma: Referencia a la Teoría de Hajek

La tabla siguiente muestra las funciones posibles a usar como T-Norma y S-Norma: Puede observarse que entre las funciones T-normas definidas están las definidas por Hajek para CTR (*) y CONJ (min"). Por su parte, entre las S-normas están las definidas por Hajek para DISY (max) y GLOB (Prospector).

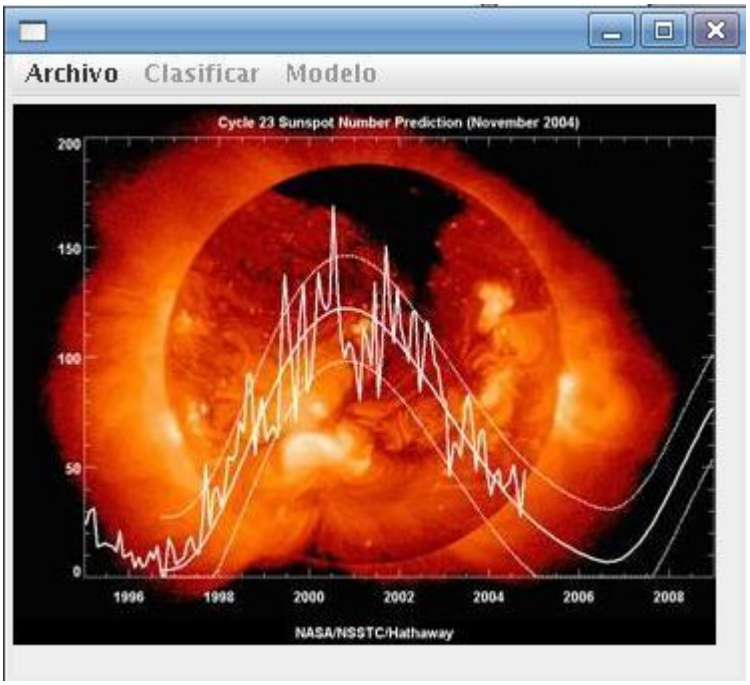
T-norma(x, y)	**S-norma(x, y)**	**Nombre del par T-S**
min(x, y)	max(x, y)	Mínimo-Máximo
x * y	x + y - x*y	Probabilístico
max(0, x+y-1)	min(1, x+y)	Lukasiewicz

En el modelo difuso, las funciones anteriores se usan de la manera expresada en la tabla siguiente. Esto quiere decir que para realizar los procesos de CTR y CONJ se puede usar cualquiera de las funciones T-norma, mientras que para los procesos de DISY y GLOB se puede usar cualquiera de las funciones S-norma. Para la NEG se debe usar la función NEG(X) = 1 – X.

T-norma(x, y)	**S-norma(x, y)**
CTR y CONJ	DISY y GLOBAL

Anexo#3

Interfaces de Usuario



Interfaz de Crear Modelo

Interfaz de Contarse a la Base de Datos

Interfaz de Leer de la Base de Datos

Interfaz de Realizar Predicción

Anexo#4

Descripción de las clases.

Nombre: CI_RealizarPrediccion	
Tipo de clase: Interfaz	
Atributo	**Tipo**
cargarButton	jButton
txfFichero	JTextField
prediccionLabel	JLabel
modeloButton_1	JButton
txfModelo	JtextField
prediccionButton	JButton
cancelarButton	JButton

motor	MotorInferencia
buffer	BufferedReader
cantcont	int
Para cada responsabilidad:	
Nombre:	CI_RealizarPrediccion()
Descripción:	Construye la clase, realizando todas las operaciones definidas.
Nombre:	cargar(String direccion)
Descripción:	Carga el fichero de la molécula a predecir.
Nombre:	cargarmolecula()
Descripción:	Lee el fichero de la molécula y almacena los datos.
Nombre:	cargarmodelo()
Descripción:	Lee el fichero del modelo y almacena los datos
Nombre:	cargamod(String direccion)
Descripción:	Carga el fichero del modelo a usar para la predicción.

Nombre: CC_MotorInferencia	
Tipo de clase: Control	
Atributo	**Tipo**
molecula	MolPredecir
conjDifuso	ArrayList<ConjuntoDifuso>
Valorxparticion	ArrayList<PertenenciaXParticiones>
conjunciones	ArrayList<Double>
CTR	ArrayList<Double>
glob	ArrayList<GLOB>
reglas	ArrayList<Regla>
m	int
ptsAcotamiento	ArrayList<Double>
valordefusificado	(Double)
clasifnum	Double
clasiflit	String
mivariable	Variable
pto0	Double
ptoa	Double
ptob	Double

ptof	Double
sepuedeclasif	boolean
Para cada responsabilidad:	
Nombre:	CC_MotorInferencia()
Descripción:	Constructor de la clase.Inicializa las instancias de los objetos creados.
Nombre:	getMolecula()
Descripción:	Devuelve los datos de la molécula a la que se le hará la predicción.
Nombre:	CalcularPert()
Descripcion:	Busca para cada regla y para cada valor de los descriptores de la molécula, la pertenencia que tiene para todos los conjuntos difusos en cada variable de la regla.
Nombre:	ValorXParticion(Double x, String variable)
Descripción:	Es usado por el método anterior para dado el valor de un descriptor de la molécula y la variable de una regla determinada, buscar el valor de pertenencia a cada partición.
Nombre:	ConjuncionXRegla()
Descripción:	Calcula la conjunción por cada regla.
Nombre:	CTRxRegla()
Descripción	Calcula la contribución por cada regla.
Nombre:	CalcularGLOB()
Descripción:	Calcula el Glob por consecuente de las reglas.
Nombre:	getRegla()
Descripción:	Devuelve las reglas del modelo difuso.
Nombre:	getConjDifuso()
Descripción:	Devuelve los conjuntos difusos del modelo.
Nombre:	calcularPtosAcotamiento(Double p0, Double a, Double b, Double pf)
Descripción:	Dados los ptos críticos de la función de la Actividad, se determina la función de pertenencia acotando con los valores del Glob.
Nombre:	Clasificar(Double valormol)
Descripción:	Calcula los valores de μ para cada $x_{i,}$ usando la función de pertenencia de la Actividad.
Nombre:	Defusificar()
Descripción:	Utiliza Clasificar(Double) para calcular los μ(s) y guardarlos en mivariable. Utiliza también Defuzzify() para clacular el Cog.
Nombre:	getpto0()
Descripción	Devuelve el primer punto crítico de la función de pertenencia de la Actividad.
Nombre:	getptoa()
Descripción:	Devuelve el segundo punto crítico de la función de pertenencia de la Actividad.

Nombre:	getptob()
Descripción:	Devuelve el tercer punto crítico de la función de pertenencia de la Actividad.
Nombre:	getptof()
Descripción:	Devuelve el cuarto punto crítico de la función de pertenencia de la Actividad.
Nombre:	getsepuedeclasif()
Descripción:	Devuelve false si el Glob dio 0 y no se puede acotar la función de pertenencia de la Actividad, sino devuelve true.

Nombre: CE_Regla	
Tipo de clase: Entidad	
Atributo	**Tipo**
varXAnte	ArrayList<String>
Antecedentes	ArrayList<String>
Consecuente	String
Relevancia	Double
Para cada responsabilidad:	
Nombre:	CE_Regla()
Descripción:	Construye la clase inicializando los atributos.
Nombre:	setantecedente(int pos, String ante)
Descripción:	Modifica el antecedente en la posoción dada.
Nombre:	setvarXAnte(int pos, String var)
Descripción:	Modifica el valor de cada antecedente.
Nombre:	setconsecuente(String cons)
Descripción:	Modifica el consecuente de la regla.
Nombre:	setrelevancia(Double rel)
Descriptor:	Modifica el valor de la relevancia de la regla.
Nombre:	getAntecedente(int ps)
Descripción:	Devuelve el antecedente en la posición especificada.
Nombre:	getVarXAnte(int pos)
Descripción:	Devuelve el valor del antecedente en la posición especificada.
Nombre:	getAntecedenttes()
Descripción:	Devuelve todos los antecedentes de la regla.
Nombre:	getrelevancia()
Descripción:	Devuelve el valor de relevancia de la regla.
Nombre:	getConsecuente()
Descripción:	Devuelve el consecuente de la regla.

Nombre:	getVarXAntecedente()
Descripción:	Devuelve los valores de los antecedentes.

Nombre: CE_MolPredecir	
Tipo de clase: Entidad	
Atributo	**Tipo**
idmolecula	String
valoresDesc	ArrayList<Double>
variables	ArrayList<String>
Para cada responsabilidad:	
Nombre:	CE_MolPredecir()
Descripción:	Constructor de la clase.
Nombre:	setidmolecula(String nombre)
Descripción:	Modifica el nombre de la molécula.
Nombre:	setValoresDesc(int pos, Double val)
Descripción:	Adiciona los valores de los descriptores.
Nombre:	setvariables(int pos, String nombre)
Descripción:	Adiciona las variables que identifican los descriptores.
Nombre:	getvariables()
Descriptores:	Devuelve todas las variables.
Nombre:	getValoresDesc()
Descripción:	Devuelve los valores de los descriptores.
Nombre:	getNombre()
Descripción:	Devuelve el atributo idmolecula.

Nombre: CE_ConjuntoDifuso	
Tipo de clase: Entidad	
Atributo	**Tipo**
IdPart	String
IDvar	String
pto1	Double
pto2	Double
pto3	Double
pto4	Double
conjuntoIzquierda	int
conjuntoDerecha	int
Para cada responsabilidad:	

Nombre:	CE_ConjuntoDifuso(String part,String s, Double p1, Double p2, Double p3, Double p4, int izq, int der)
Descripción:	Constructor de la clase.
Nombre:	getIDvar()
Descripción:	Devuelve el valor del atributo Idvar.
Nombre:	getIDPart()
Descripción:	Devuelve el valor del atributo IdPart.
Nombre:	getpto1()
Descripción:	Devuelve el valor del atributo pto1.
Nombre:	getpto2()
Descripción:	Devuelve el valor del atributo pto2.
Nombre:	getpto3()
Descripción:	Devuelve el valor del atributo pto3.
Nombre:	getpto4()
Descripción:	Devuelve el valor del atributo pto4.

Nombre: CE_GLOB	
Tipo de clase: Entidad	
Atributo	**Tipo**
clase	String
valor	Double
Para cada responsabilidad:	
Nombre:	CE_GLOB(String s, Double val)
Descripción:	Contruye pasando los valores que tomarán los atributos inicialmente.
Nombre:	CE_GLOB()
Descripción:	Contruye sin parámetros.

Nombre: CE_PertenenciaXParticiones	
Tipo de clase: Entidad	
Atributo	**Tipo**
IDVar	String
IDPart	String
Pertenencia	Double
Para cada responsabilidad:	
Nombre:	CE_PertenenciaXParticiones()
Descripción:	Constructor de la clase.

Nombre: CE_ Variable	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre	String
min	Double
paso	Double
Mius	ArrayList<Double>
Para cada responsabilidad:	
Nombre:	CE_Variable()
Descripción:	Contructor de la clase
Nombre:	addValor(int pos, Double val)
Descripción:	Adiciona valores a Mius

Nombre:	getMius()
Descripción:	Devuelve los valores de μ(xi)
Nombre:	getMIn()
Descripción:	Devuelve el valor de min.
Nombre:	getPaso()
Descripción:	Devuelve el valor de paso.

Nombre: CI_NauckUCI	
Tipo de clase: Interfaz	
Atributo	Tipo
JMenuArchivo	JMenu
JMenu2	JMenu
JMenuClasificar	JMenu
JMenuModelo	JMenu
JMenuBar1	JMenuBar
JMenuBar2	JMenuBar
menuItemConectDataBase	JMenuItem
menuItemLoadFromDataBase	JMenuItem
menuItemLoadFromFile	JMenuItem
menuItemLoadExit	JMenuItem
menuItemCreateRuleBase	JMenuItem
menuItemPruneRuleBase	JMenuItem
menuItemCreateAndPrune	JMenuItem
menuItemClassifyData	JMenuItem
menuItemRuleExport	JMenuItem
menuItemStopProcess	JmenuItem
menuItemCloseProject	JmenuItem
conect	boolean
readFile	boolean
lSymAction	SymAction
tran	Transacciones
progressDialog	DialogProgress
dataTable	TablaDeDatos
appData	TablaDeDatos
nefclass	GestionarModelo
parameter	ListaDeParametros
newProject	boolean

projectSaved	boolean
message	DialogMessage
fsLearning	RuleLearningProgressDialog
functionPanel	FunctionLayoutPanel
loggin	boolean
logFile	PrintWriter
training	FrameStatistics
projectSpecifications	DialogProjectSpecification
fuzzySetsTrained	boolean
statistics	FrameStatistics
appStatistics	FrameStatistics
application	FrameStatistics
fuzzySets	FrameShowFS
Para cada responsabilidad:	
Nombre:	CI_NauckUCI()
Descripción:	Constructor de la clase
Nombre:	SymAction()
Descripción:	Lleva según los eventos en el menu a los métodos respectivos
Nombre:	menuItemCreateAdPrune_ActionEvent(ActionEvent action)
Descripción:	Usando el atributo nefclass crea y poda la base de reglas
Nombre:	ShowTrainingFrames()
Descripción:	Muestra las ventanas de entrenamiento.
Nombre:	menuItemPruneRuleBase_ActionPerformed(ActionEvent action)
Descripcion:	Se encarga de podar la Base de Reglas.
Nombre:	ShowFuzzySetLearningFrame()
Descripción:	Muestra la ventana de los conjuntos difusos.
Nombre:	menuItemLoadFormDataBase_ActionPerformed(ActionEvent action).
Descripción:	Carga los datos de las moleculas de la Base de Datos.
Nombre:	SetupNefclassInterface()
Descripción:	Muestra las ventanas de progreso del entrenamiento.
Nombre:	readDataFile(String s, boolean flag)
Descripción:	Lee los datos del fichero.
Nombre:	menuItemCloseProject_ActionPeformed(ActionEvent action).
Description:	Permite cerrar el proyecto iniciado.
Nombre:	checkRunning()
Descripción:	Verifica si el proyecto está corriendo.
Nombre:	closeAllNefclassFrames().
Descripción:	Cierra todas las ventanas del proyecto.

Nombre:	menuItemNewProject_ActionPerformed(ActionEvent action)
Descripción:	Permite que se ejecute un nuevo proyecto.
Nombre:	menuItemStopTraining_ActionPerformed(ActionEvent action)
Descripción:	Detiene el proceso de entrenamiento de la base de reglas.
Nombre:	menuItemRulesExport_ActionPerformed(ActionEvent action)
Descripción:	Permite exportar el modelo a un fichero .dat.
Nombre:	tiStringFS()
Descripción:	Convierte a un String los datos de los conjuntos difusos
Nombre:	menuItemConectDataBase_ActionPerformed(ActionEvent action)
Descripción:	Muestra la ventana que permite la conección a la base de datos.
Nombre:	IsConect()
Descripción:	Conecta a la Base de Datos.
Nombre:	menuItemLoadFromFile_ActionPerformed(ActionEvent action)
Descripción:	Muestra la ventana que nos permite subir los datos desde un fichero.
Nombre:	showRuleLearninFrame()
Descripción:	Muestra la ventana que contiene el progreso del aprendizaje de las reglas.
Nombre:	menuItemCreateRuleBase_ActionPerformed(ActionEvent action)
Descripción:	Permite crear la Base de Reglas.
Nombre:	menuItemClassifyData_ActionPerformed(ActionEvent Action)
Descripción:	Clasifica los datos usados según el modelo creado.
Nombre:	menuItemExit_ActionPerformed(ActionEvent action)
Descripción:	Cierra la aplicación.
Nombre:	setTran(Transacciones tran)
Descripción:	Modifica las transacciones.

Nombre: CI_EspecificacionesDelModelo	
Tipo de clase: Interfaz	
Atributo	**Tipo**
message	DialogMessage
dataTable	TablaDeDatos
nefclass	GestionarModelo
parameter	ListaDeParametros
nefclassWasNull	boolean
fComponentsAdjusted	boolean
tabPanelProjectSpecification	TabPanel
panelProject	Panel
panelData	Panel

wrappingLabelDataFileName	WrappingLabel
textFieldDataFileName	TextField
labelButtonDataFileOpen	Button
panelClassifier	Panel
wrappingLabelClassifierFussysets	WrappingLabel
radioButtonClassificationFSSame	Checkbox
labelClassifierFSForm	Label
labelButtonClassifierFSSpecify	Button
panelRuleCreation	Panel
wrappingLabelSizeRB	WrappingLabel
radioButtonGroupPanelSizeRB	RadioButtonGroupPanel
radioButtonMaxNumberFR	Chackbox
radioButtonAutomaticFR	Checkbox
textFieldSizeRB	TextField
panelTrainingControl	Panel
wrappingLabelStopControl	WrappingLabel
panelStopControl	Panel
labelNoEpochsMax	Label
textFieldNoEpochsMax	TextField
labelNoEpochsMin	Label
textFieldNoEpochsMin	TextField
labelButtonProjectSpecificationOK	Button
labelButtonProjectSpecificationCancel	Button
Para cada responsabilidad:	
Nombre:	CI_EspecificacionesDelModelo(Frame frame, boolean bol)
Descripción:	Constructor de la clase.
Nombre:	setDataTable(TableDeDatos datos)
Descripción:	Entra los nuevos datos entrados en la estructura.
Nombre:	setNefclass(GestionarModelo nefclass)
Descripción:	Entra los datos a la instancia del objeto GestionarModelo
Nombre:	setParameter(ListaDeParametros parameter)
Descripción:	Entra a la estructura los datos necesarios
Nombre:	getDataTable()
Descripción:	Devuelve los datos del atributo dataTable.
Nombre:	getNefclass()
Descripción:	Devuelve los datos del atributo nefclass.
Nombre:	updateParameterOK()
Descripción:	Actualiza los datos con los nuevos parámetros especificados.

Nombre: CI_DataBase	
Tipo de clase: Interfaz	
Atributo	**Tipo**
jButton1	JButton
jButton2	JButton
jLabel1	JLabel
jLabel2	JLabel
jLabel3	JLabel
jLabel4	JLabel
jLabel5	JLabel
jLabel6	JLabel
jPasswordField1	JPasswordField
jTextField1	JTextField
jTextField2	JTextField
jTextField3	JTextField
jTextField4	JTextField
Para cada responsabilidad:	
Nombre:	CI_DataBase(int i, int j, NauckUCI owner)
Descripción:	Constructor de la clase.
Nombre:	getT()
Descripción:	Devuelve la transacción.
Nombre:	initComponents()
Descripción:	Usado por el contructor para inicializar los atributos.
Nombre:	JButton1ActionPerformed(ActionEvent action)
Descripción:	Se crea la transacción y se concta a la base de datos.
Nombre:	conectar()
Descripción:	Conecta a la base de datos.
Nombre:	getTransacciones()
Descripción:	Devuelve las transacciones hechas.
Nombre:	getDataServer()
Descripción:	Devuelve el nombre del Servidor de Base de datos.
Nombre:	getPort()
Descripción:	Devuelve el puerto por el que se conecta a la Base de datos
Nombre:	getUser()
Descripción:	Devuelve el usuario.
Nombre:	getPassword()
Descripción:	Devuelve la contraseña de acceso.
Nombre:	FormWindowClosing(ActionEvent action)
Descripción:	Cierra la ventana de la conección a la base de datos.

Nombre: CC_GestionarModelo	
Tipo de clase: Control	
Atributo	**Tipo**
description	String
classNaes[]	String
maxrules	int
dataTable	TablaDeDatos
applicationData	TablaDeDatos
ruleBaseCopy	Vector
ConsistencyCheckMessage	String
ruleBaseBackup	Vector
partitions[]	ParticionDifusa
bestPartitions[]	ParticionDifusa
initPartitions[]	ParticionDifusa
savePartitions[]	ParticionDifusa
useNamesfromData	boolean
saved	boolean
ruleBaseCreateMethod	int
ruleBaseSizeAutomatic	boolean
relearnRuleBase	boolean
tNorm	TNorm
selected	int
logFile	PrintWriter
logging	boolean
statusArea	TextArea
ruleStatusArea	TextArea
fsStatusArea	TextArea
resultTextArea	TextArea
pbar1	ProgressBar
pBar2	ProgressBar
pBar3	ProgressBar
pBar4	progressBar
flPanel	FunctionLayoutPanel
errorGraph	FunctionPanel
learningThread	Thread
pruningThread	Thread
running	boolean

confusionMatrix[][]	int
unclassified	int
misclassified	int
error	double
estimatedError	double
estimatedErrorStdError	double
trainingDataIsClassified	boolean
validationDataIsClassified	boolean
completeDataTableIsClassified	boolean
applicationDataIsClassified	boolean
validationModeIsSet	boolean
performancesComputed	boolean
validationMode	int
validationPart	int
learningRate	double
misclassiPercent	double
misclassifications	int
stopError	double
KeepOrder	boolean
mustOverLap	boolean
asymmetric	boolean
addUpToOne	boolean
useWeights	boolean
batchLearning	boolean
minOnly	boolean
invokeRuleLeraning	boolean
invokeFuzzySetLearning	boolean
invokePruning	boolean
invokeInternalPruning	boolean
externalStop	boolean
haseCategoricalVariables	boolean
buffer	BufferedReader
tokenizer	streamTokenizer
actionListener	ActionListener
Para cada responsabilidad:	
Nombre:	CC_GestionarModelo(int a, int b, int c, int d)
Descripción:	Construcctor de la clase
Nombre:	createAndPrune()

Descripción:	Crea las reglas y las poda
Nombre:	getNumberOfRules()
Descripción:	Devuelve la cantidad de reglas creadas.
Nombre:	automaticPruning()
Descripción:	Poda las reglas automaticamente despues de ser creadas.
Nombre:	usesData(TablaDeDatos dataTable)
Descripción:	compruba que los datos que se van a procesar son los mismos que ya están almacenados
Nombre:	setData(TablaDeDatos dataTable)
Descripción:	Modifica los datos guardados.
Nombre:	isRunning()
Descripción:	Dice si está procesando los datos.
Nombre:	stopTraining()
Descripción:	Detiene el proceso de entrenamiento.
Nombre:	getDimofpartitions()
Descripción:	Devuelve la dimensión de las particiones.
Nombre:	getPartitions()
Descripción:	Devuelve las particiones creadas.
Nombre:	learning(boolean flag1, boolean flag2)
Descripción:	Ejecuta el proceso de aprendizaje de las reglas.
Nombre:	classifyData(TextArea text)
Descripción:	Clasifica los datos en dependencia del modelo que se crea.
Nombre:	computeRulePerformance(Vector vector)
Descripción:	Calcula la relevancia de cada regla.
Nombre:	classifyTrainingAndValidationData(BufferedWriter buffer)
Descripción:	Escribe las clasificaciones de los datos de entrenamiento.
Nombre:	classifyAll(TablaDeDatos tabla, BufferedWriter buffer)
Descripción:	clasifica los datos entrados y los escribe en el buffer.
Nombre:	classify(TablaDeDatos tabla,boolean flag1, boolean flag2,int a, BufferedWriter buffer)
Descripción:	Usado por classifyAll para clasificar uno a uno.
Nombre:	createPartitions(int i, int j)
Descripción:	Crea las particiones difusas para cada variable.
Nombre:	resetConfusionMatrix()
Descripción:	Pone los valores de la matriz en 0.

Nombre: CE_Transacciones	
Tipo de clase: Entidad	
Atributo	**Tipo**
usuario	String
pass	String
basedatos	String
ubicación	String
con	Connection
stmt	Statement
porCiento	int
numPatActivo	int
numPatInactivo	Int
limitInf	int
limitSup	int
inActivo	boolean
Para cada responsabilidad:	
Nombre:	setPorCiento(int p)
Descripción:	Modifica el porciento de datos a procesar.
Nombre:	Transacciones(String bd, String ubicac,String user, String pass)
Descripción:	Constructor de la clase.
Nombre:	conectar()
Descripción:	Realiza la conección
Nombre:	getNumPatActivo()
Descriptción:	Devuelve la cantidad de patrones activos que se entraron.
Nombre:	setRangos(ArrayList<Double> min, ArraylList <Double> max)
Descripción:	Modifica los rangos pasados los nuevos parámetros.
Nombre:	executeQuery(int fila, int cant)
Descripción:	Ejecuta la vista determinada para prodesar los datos..

GLOSARIO

Actividad Biológica: Es la actividad biológica de una molécula.

Compuestos orgánicos: Son sustancias químicas basadas en cadenas de carbono e hidrógeno. En muchos casos contienen oxígeno, y también nitrógeno, azufre, fósforo, boro y halógenos.

Descriptor: Número que caracteriza estructuralmente la molécula.

Entrenamiento: Acción que se realiza para el aprendizaje de las neuronas.

Especialista Químico: Cliente que utilizará el sistema.

Índices híbrido: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad en el grafo completo donde los vértices de esa matriz de conectividad se ponderan con el valor de una propiedad químico-física del tipo de átomo que separan.

In silico: Está es una expresión usada para significar “realizado en la computadora o vía la “simulación de computadora que modelan procesos naturales o del laboratorio”.

Índices topológico: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad del grafo molecular desprovisto de hidrógeno.

Metodología: Define quién hace qué, cómo y cuando.

Modelo: Representación abstracta de la realidad.

Modelo difuso:Modelo que se crea a partir del las reglas difusas generadas.

Predicción: Acción que el sistema realiza para emitir un resultado.

Software: Termino genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.

Trabajador.: Persona que se encarga de generar los nuevos modelos y realizar el entrenamiento de los mismos.