UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD #6

TÍTULO: "alasGRATO: DESARROLLO DEL MÓDULO DE CLASIFICACIÓN Y REGRESIÓN UTILIZANDO MÁQUINAS DE SOPORTE VECTORIAL."

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

AUTORES:

LUIS GRABIEL GARCÍA NÁPOLES

YANET MEDINA PATRICIO

TUTORES:

DR. RAMÓN CARRASCO VELAR

MSC. AURELIO ANTELO COLLADO

MSC. YAIKIEL HERNÁNDEZ DÍAZ

CIUDAD DE LA HABANA, CUBA

JUNIO, 2010

"¿Para qué, sino para poner la paz entre los hombres, han de ser los adelantos de la ciencia?"

José Martí

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 _____.

_____ _____

Luis Grabiel García Nápoles (Autor) Yanet Medina Patricio (Autor)

_____ _____

Dr. Ramón Carrasco Velar (Tutor) Msc. Aurelio Antelo Collado (Tutor)

_____

Msc. Yaikiel Hernández Díaz (Tutor)

Datos de Contactos

Autores:

Luis Grabiel García Nápoles **email: lggarcia@estudiantes.uci.cu**

Yanet Medina Patricio **email: ypatricio@estudiantes.uci.cu**

Tutores:

Ramón Carrasco Velar:

Doctor en Ciencias Químicas, profesor asistente.

email: rcarrasco@uci.cu

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

Aurelio Antelo Collado**:**

Máster en Matemática Aplicada, Ingeniero Industrial, profesor asistente.

email: aantelo@uci.cu

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

Yaikiel Hernández Díaz**:**

Ingeniero en Ciencias Informáticas, profesor instructor.

email: yhernandezd@uci.cu

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

Agradecimientos

Quiero agradecer, a mi madre que con apenas 15 años pudo ser madre, y cuando digo madre no lo digo precisamente porque me tuvo sino porque supo ser madre desde el momento en que nací. Ser madre y padre a la vez, es tarea difícil, ¿Cómo pudo? ¡Increíble!. Si hoy estoy donde estoy y soy lo que soy se lo debo a ella, ¡todo!. Para decirte lo tanto que te quiero, palabras, no tengo.

Mi padrastro ha sido ha sido unas de las personas más correctas con las que he interactuado en mi vida; profesional y buen padre, sí... buen padre. La facultad de verlo como tal, es concedida a todo aquel que tenga ojos. Observar no es mirar, es ver más allá de ese horizonte limitado que divisa nuestros ojos. Gracias por todo, que no es poco.

Mi tía Mayuli, no tengo cómo agradecer su existencia y todo lo que ha representado para mi en todo estos años, ha sabido escucharme y comprenderme, sobre todo comprenderme. Sin temor a que exista un sorbo de equivocación digo: Eres mi madre también.

Mi "joven" abuela Acela, le tengo que agradecer mucho, mucho. No dejó de esforzarse nunca para que yo siguiera este camino por el que voy.

Al Dr. Fidel Moras, por ser un ejemplo de un profesional de excelencia, por todas las charlas "estratégicas". Siempre será parte de mi familia, siempre.

Mi abuela Elida, mis tíos Manolo, Orlando y Luis que nunca pensaron dos veces brindarme su ayuda, a mi tía Acelita por ser tan cariñosa, a todo ellos que de una forma u otra me ayudaron, gracias.

Un agradecimiento especial a mi tío José Eduardo -Pepe- por todo lo que me ha ayudado en todos estos años, nunca lo voy a olvidar. Una persona que admiro muchísimo. Gracias, te debo mucho.

Agradecer a mi tutor el Dr. Carrasco por su profesionalismo, por todos los consejos que me ha dado, y todas sus provechosas anécdotas. Hemos hablado durante horas pero en cada una de esas horas me ha hecho vivir unos años.

A mi tutor Yaikiel por comprender las situaciones difíciles y ser además, amigo.

Agradecimientos

Agradezco mucho a Aurelio, no sólo porque también es mi tutor y desde tan lejos siempre estuvo dispuesto a ayudarme, sino porque desde que trabajamos juntos ha sido siempre una persona correcta y por haber sido flexible conmigo en muchas ocasiones. Espero no haberlo defraudado.

Agradecer a todas las personas que consulté para realizar mi trabajo. Liusmila y Alberto, gracias.

Mis compañeros Dayron, Andrés y Feliz gracias por soportarme incondicionalmente.

Luis Grabiel García Nápoles

Agradecimientos

Quisiera agradecer a mis padres por brindarme su apoyo incondicional cada vez que lo necesité, por confiar siempre en mí y en mis decisiones, por sus buenos y siempre oportunos consejos, por inculcarme un grupo de valores sin los cuales hoy no sería quien soy. Sin ellos no hubiera llegado hasta aquí.

A mi papá por ser la luz que me mostró el camino, por ser tan exigente y cada vez exigirme más, demostrándome que siempre se puede dar más.

A mi mamá por ser más que mi madre, mi amiga, por darme y enseñarme la vida. Por ser mi inspiración, un motivo por el cual esforzarme cada día más. Por el sacrificio que hizo para que yo terminara mis estudios, por estar conmigo en las buenas y en las malas.

A mi hermana por ser única, espléndida, por impulsarme y ayudarme tanto.

A Eric y familia por acogerme tantos años como un miembro más, por dar tanto por mí sin interés alguno. Por ser mi apoyo la mayor parte de mi carrera, por hacer más alegres cada uno de los días vividos en esta escuela. Decirle que nunca lo olvidaré.

A Aurelio por confiar en mí y darme la seguridad de confiar en él. Por estar ahí cada vez que necesité su ayuda. Por hacer menos compleja la etapa de licencia, por sus consejos, sus charlas, por tantos ratos inolvidables. Por tenerme siempre presente. Alguien que ha hecho por mí, cosas inolvidables. De verdad lo admiro y lo respeto mucho.

A mi compañero de tesis Luis G, por ser tan optimista, por ser alguien excepcional desde el tercer años y hasta hoy. Por su incondicionalidad, porque junto a él logré aprender muchas cosas. Desearle mucha suerte en su vida y decirle que siempre lo tendré presente.

A mis tutores por el conocimiento que han transmitido, por guiarme en este difícil camino.

A todos mi amigos y compañeros, a los viejos y nuevos conocidos. Cada uno de ellos formó un hito en mi vida y en mi carrera. Por llenar cada uno de los recuerdos que me llevo. Especialmente a Alberto, las muchachitas de la secretaría, a Fifi por ser amiga, más que eso como una madre para mí.

A Osmay y no por último menos importante, por haber llegado a mi vida y compartir conmigo mis últimos momentos en la UCI, por enseñarme que las personas no son como uno quiere que sea, por entenderme

como nadie y soportar cada una de mis pesadeces. Agradecerle su ayuda, apoyo y cariño. Decirle que a pesar de los momentos felices y tristes que hemos tenido, lo quiero mucho.

A todos muchas gracias.

Yanet Medina Patricio

Dedicatoria

De Luis G:

A mi madre Reina Luisa, si nunca he pensado dejar de luchar, ha sido por ella.

A mi tía Mayuli por su amor incondicional, y todo su apoyo.

A mi abuela Acela, por su cariño.

A toda mi familia que siempre me apoyó.

De Yanet:

Dedico mi trabajo de diploma a mis padres (Juana y Romelio), quienes me dieron la vida e hicieron posible el éxito de mi vida futura.

A mi hermana Maritriny que a pesar de la distancia siempre ha estado pendiente de mí.

A mis abuelitos que se que me miran desde algún lado y que están orgullosos de mi.

A mi padrastro Manolo, por el apoyo incondicional que recibí de él, por preocuparse por mí y mis estudios en todo momento. Sinceramente sin él muchas cosas no hubieran sido posibles.

A toda mi familia y a todos aquellos que de alguna forma u otra me ayudaron y depositaron toda su confianza en mí.

A todas esas personas que no aparecen aquí y forman parte de mi vida y mi trayectoria

A los que de cualquier manera forman parte de mi éxito.

Resumen

El presente trabajo está enmarcado dentro del proyecto Plataforma para la Predicción de Actividad Biológica de Compuestos Orgánicos (alasGRATO) llevado a cabo por el grupo de Bioinformática de la Universidad de las Ciencias Informáticas. Es un proyecto concebido en módulos, uno de cuyos módulos consiste en la predicción de actividad biológica de compuestos orgánicos que incluye Máquinas de Soporte Vectorial como técnica de inteligencia artificial.

El descubrimiento de nuevas moléculas de interés farmacéutico está estrechamente relacionado con la búsqueda de información en grandes bases de datos, es por eso que el desarrollo de aplicaciones informáticas para extraer conocimiento de la información generada en los laboratorios y luego ser almacenada en una base de modelos generados, se manifiesta como una de las necesidades importantes en la Bioinformática. En el presente trabajo se muestran los resultados del análisis, diseño, implementación y validación de los modelos de las máquinas de soporte vectorial para el desarrollo de modelos de regresión y clasificación.

El objetivo de la investigación es desarrollar un módulo de predicción de actividad biológica para la plataforma alasGRATO que utilice una base de modelos generados para los distintos ensayos farmacológicos con Máquinas de Soporte Vectorial como técnica de inteligencia artificial y los diferentes kernels de procesamiento, como son el lineal, el RBF, el polinomial y el sinusoidal. Se muestra la implementación de los servicios web y las librerías correspondientes para conectarse a los mismos.

Palabras Claves: *Base de conocimientos, Clasificación, Kernel, Máquina de Soporte Vectorial, Modelo, Regresión*

INDICE

INTRODUCCIÓN

El presente trabajo surge en el marco del proyecto ¨Plataforma inteligente para la predicción de Actividad Biológica de Compuestos Orgánicos¨ alasGRATO, llevado a cabo por la línea de investigación de Diseño de Fármacos perteneciente al grupo de Bioinformática de la Universidad de las Ciencias Informáticas.

La predicción de la actividad biológica de compuestos orgánicos es una línea fundamental en la Industria Médico Farmacéutica Mundial. Es innumerable la cantidad de compuestos que se conocen hoy a nivel internacional y el costo de los ensayos necesarios para comprobar la utilidad de uno en específico, es elevado, pues solo un 5 ó 10% de las moléculas que llegan a fase de ensayos clínicos terminan siendo comercializadas. Las técnicas de la computación han sido una vía de solución que permite acelerar el proceso de investigación-desarrollo de nuevos fármacos y disminuir los costos. [1]

Específicamente para la plataforma alasGRATO, desde sus inicios, la actividad fundamental estaba enfocada en predecir actividad biológica. Se llevaron a cabo investigaciones donde se aplicaron diversas técnicas de inteligencia artificial con el objetivo de conocer cuáles eran los resultados de la predicción de actividad biológica de compuestos orgánicos ante el procesamiento de grandes volúmenes de datos. Además se efectuaron tesis de grados con las cuales se pretendía mejorar las predicciones arrojadas por cada una de las técnicas aplicadas.

Actualmente la plataforma alasGRATO cuenta con una base de datos de mediano tamaño, donde se almacenan descriptores moleculares que se utilizan para el establecimiento de modelos de relación estructura–actividad para la predicción de actividad biológica en los clasificadores que tienen implementados. Para lograr una buena predicción se necesita seleccionar el subconjunto de estos descriptores que aporten la información necesaria y suficiente para la generación de los modelos de clasificación y regresión en cada actividad en particular, debido a la multiplicidad de situaciones y problemas específicos.

En particular, las Máquinas de Soporte Vectorial como técnica de Inteligencia Artificial han demostrado construir buenos modelos y se han convertido en herramientas de clasificación con gran popularidad. Desde que Vladimir Vapnik propuso los primeros algoritmos de Máquinas de Soporte Vectorial (utilizados actualmente) se ha mostrado la capacidad de construir modelos precisos con una relevancia práctica para la clasificación y regresión. [2]

Introducción

Es una necesidad de la plataforma alasGRATO guardar los resultados del entrenamiento de los distintos ensayos presentes en su base de datos, en modelos de clasificación y regresión, dichos modelos serían almacenado en una base de modelos generados, aplicando para la generación de los modelos una técnica de Inteligencia Artificial (IA) de probada eficiencia como son, las Máquinas de Soporte Vectorial (MSV). Lo anteriormente planteado conduce a la formulación del siguiente problema científico:

¿Cómo mejorar el módulo de predicción de actividad biológica de la plataforma alasGRATO?

Existen diferentes técnicas de IA que han sido aplicadas con éxito en la modelación de moléculas con posible actividad biológica. Las más importantes son las redes neuronales, lógica difusa, algoritmos genéticos y sus diferentes variantes. La opción propuesta en la investigación son las Máquinas de Soporte Vectorial. Por lo que se define como *objeto de estudio:*

Predicción de Actividad Biológica utilizando MSV.

Las Máquinas de Soporte Vectorial son un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel, además sirven para entrenar máquinas de aprendizaje lineal eficientemente y tanto para la clasificación como para la regresión se han encontrado muchas aplicaciones como clasificación de imágenes, reconocimiento de caracteres, detección de proteínas, clasificación de patrones, identificación de funciones, etc. Por lo que se formula como *Campo de Acción:*

Predicción por clasificación y regresión de Actividad Biológica en Compuestos Orgánicos utilizando MSV.

Con el fin de solucionar el problema planteado anteriormente se define como *objetivo general:*

Desarrollar un módulo de predicción de actividad biológica para la plataforma alasGRATO con Máquinas de Soporte Vectorial como técnica de inteligencia artificial.

Para lograr dicho objetivo general se establecieron como *objetivos específicos:*

- Realizar el análisis del módulo de predicción de actividad biológica.
- Diseñar el módulo de predicción de actividad biológica.
- Implementar el Módulo de predicción de actividad biológica.

- Crear la base de modelos para los distintos ensayos químicos presente en la base de datos de la Plataforma.
- Validar el Módulo de predicción de actividad biológica.

Como *aporte práctico* se espera un módulo de predicción de actividad biológica para la plataforma alasGRATO que utilice una base de modelos generados para los distintos ensayos químicos con Máquinas de Soporte Vectorial como técnica de inteligencia artificial. Además de contar con la funcionalidad de selección de variables para eliminar las variables con información redundante.

El presente documento consta de Resumen, Introducción y cuatro capítulos que constituyen el cuerpo de la tesis, Conclusiones, Recomendaciones, Referencias bibliográficas, Anexos y Glosario de Términos. Los capítulos son:

Capítulo 1: Fundamentación Teórica.
En este capítulo se presenta un resumen de la investigación más actual sobre las Máquinas de Soporte Vectorial así como sus principales tipos y definiciones. Se destacan las tendencias actuales y el estado del arte a tener en cuenta. Se describen las aplicaciones informáticas más relevantes desarrolladas hasta el momento sobre este tema y los autores que la desarrollan.

Capítulo 2: Características del sistema.
En este capítulo se describen las principales características del sistema a desarrollar y se muestra el modelo de dominio. Se definen los requisitos funcionales, los requisitos no funcionales y los actores que intervienen en el sistema. Se presenta además el diagrama de casos de uso del sistema, junto con una breve descripción de los casos de uso identificados.

Capítulo 3: Diseño del Sistema.
En este capítulo se brinda una descripción del estilo arquitectónico utilizado para el desarrollo del sistema, los patrones de diseño empleados y los diagramas de clases de diseño e interacción de los casos de uso principales. Se muestra también el diagrama de despliegue del sistema.

Capítulo 4: Implementación y Pruebas del Sistema.

Introducción

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los principales casos de uso.

CAPITULO 1 FUNDAMENTACION TEORICA

Introducción

En este capítulo se realiza y se muestra un estudio acerca del desarrollo de las Máquinas de Soporte Vectorial. Además se hace una breve valoración de las técnicas, plataforma, librerías y procedimientos utilizados para dar cumplimiento al desarrollo de la plataforma. Finalmente se hace una descripción de las herramientas y tecnologías a utilizar para desarrollar la solución propuesta.

1.1 Máquinas de Soporte vectorial

El algoritmo Vector de Soporte (VS) es una generalización no-lineal del algoritmo Semblanza Generalizada, desarrollado en Rusia en los años sesenta. Está firmemente enlazado a la Teoría del Aprendizaje Estadístico [3], la cual se desarrolló a finales de la década de los 80's por Vapnik y Chervonenkis [2]. Por otra parte, esta Teoría del Aprendizaje Estadístico caracteriza propiedades del aprendizaje, habilitándole el poder de generalización de datos desconocidos. El desarrollo de los VS trae consigo el surgimiento de las Máquinas de Soporte Vectorial (MSV) [3]. Esta técnica de inteligencia artificial se enmarca dentro de las Redes Neuronales de aprendizaje supervisado, y es un clasificador eficiente cuando se desconoce la dependencia existente entre los datos y le permite la generalización de los mismos.

Las MSV pertenecen a la familia de clasificadores lineales. Mediante una función matemática denominada *kernel*, los datos originales se redimensionan para buscar una separabilidad lineal de los mismos [4].

Una característica de las MSV es que realiza un mapeo de los vectores de entrada para determinar la linealidad o no de los casos los cuales serán integrados a los multiplicadores de Lagrange para minimizar el Riesgo Empírico y la Dimensión de Vapnik-Chervonenkis [5]. De manera general, las Máquinas de Soporte Vectorial permiten encontrar un hiperplano óptimo que separe las clases.

1.1.1 Minimización del Riesgo Empírico (MRE)

Desde un punto de vista más simple, la MRE consiste en encontrar la función f(x) que minimice el riesgo promedio del conjunto de entrenamiento. En la medida en que tengamos mayor cantidad de vectores, se

puede asegurar que el riesgo empírico converge asintóticamente al riesgo esperado cuando el número de valores tiende a infinito (n->∞). [2, p. 25-29]

1.1.2 Dimensión de Vapnik-Chevonenkis (VC)

La Dimensión de Vapnik Chevonenkis es el número máximo h de vectores z1...zn que se pueden separar de todas las maneras posibles 2^h por los hiperplanos, a partir de funciones que miden el mayor número de muestras que pueden ser explicadas por el sistema. Donde para un conjunto de datos R, la familia de hiperplanos que se genera está dada por el término RD en donde D es al menos D+1 para valores de D mayor que cero [4, p. 20]. Teniendo como propiedades que:

i) La dimensión de VC h^k de cada conjunto S^k de funciones es finita. Por lo tanto,
$h_1 \leq h_2 \leq \ldots \leq h_{n\ldots}$
ii) Cualquier elemento S_k de la estructura contiene un conjunto de funciones.
La relación existente entre los conceptos anteriormente explicados viene dada por:

$$R(f) \leq R_{emp}(f) + \underbrace{\sqrt{\frac{h(\ln(2N/h)+1) - \ln(\eta/4)}{N}}}_{\text{VC confidence}}$$

Donde:
N = Número de muestras entrenadas.
R (f)= Riesgo Esperado.
Remp (f)= Riesgo Empírico.
h= Dimensión de Vapnik-Chevonenkis.

Si se realiza un análisis de la misma se puede asegurar que: el modelo más adecuado se encuentra cuando se logra un balance entre el Riesgo Empírico (Remp (f)) y la dimensión de VC, este problema se conoce como la minimización del riesgo estructural. Además, a medida que la relación N/h se hace mayor, la confianza VC se hace menor y el Riesgo Esperado se acerca al Riesgo Empírico. Una MSV mapea los puntos de entrada en un espacio de características de una dimensión mayor y encuentra el hiperplano óptimo que los separe y maximice el margen **m** entre las clases, en este espacio. **(Ver Figura 1)**

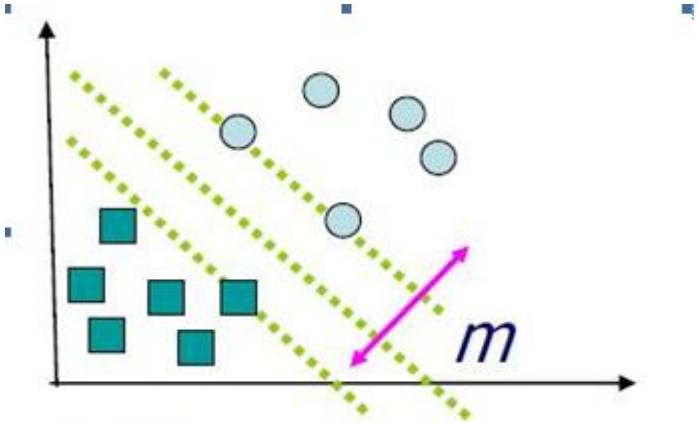

Figura 1. ¨Mapeo de los puntos de entrada¨

A partir del conjunto de entrenamiento, donde el dominio de las imágenes se encuentran en el intervalo [-1; 1], las ecuaciones de los hiperplanos H1 y H2 viene dada por: H1= wxi+b=1, H2= wxi+b=-1 y el hiperplano óptimo por wxi+b=0, partiendo de que ||w|| es la norma del vector normal al hiperplano para las clases que se representan. Por lo tanto el margen m se calcula por la expresión m=2/||w||. Maximizar dicho margen es un problema de programación cuadrática. [4]. **(Ver Figura 2)**

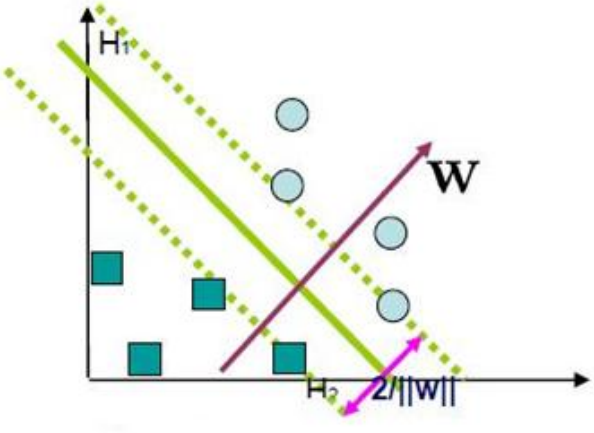

Figura 2. ¨Maximización del margen m¨

1.2 Máquina de Soporte Vectorial para Clasificación

Entre las aplicaciones más relevantes de las Máquinas de Soporte Vectorial se encuentra la Clasificación, el problema de la clasificación puede reducirse a examinar dos clases sin pérdida de generalidad. La tarea es encontrar un clasificador que funcione bien en datos futuros, es decir que generalice bien la

clasificación. [6]

Consideremos el siguiente ejemplo:

Aquí hay muchos posibles clasificadores lineales que pueden separar los datos; sin embargo solo existe un clasificador que maximiza el margen, maximiza la distancia entre él y los puntos más cercanos de los datos de cada clase. A este clasificador lineal se le denomina hiperplano separador óptimo. De forma intuitiva se puede esperar que este separador generalice bien la clasificación para datos futuros.

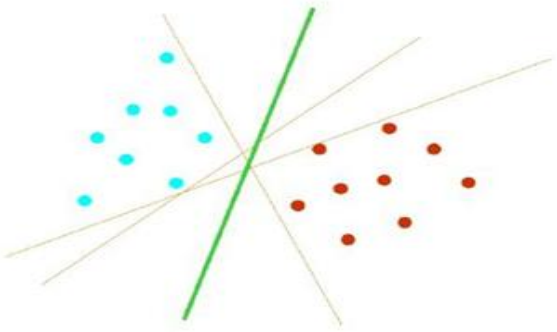

Figura 3. ¨Hiperplano separador óptimo¨

1.2.2 Tipos de Máquinas de Soporte Vectorial para la clasificación

Dentro de las máquinas de soporte vectorial para la clasificación se encuentran C_SVC, nu-SVC y one_class, la más importante para la investigación es C_SVC explicada a continuación.

Para este tipo de MSV, el entrenamiento involucra la minimización de la función error (Ec. # 1):

$$\frac{1}{2}w^t w + C\sum_{i=1}^{N}\xi_i$$

Ecuación # 1

Sujeto a las restricciones:

$$y(w^t\phi(x_i)+b) \geq 1-\xi_i \quad \text{y} \quad \xi_i \in \Re \geq 0 \text{ para } i=1,\ldots,N$$

Ecuación # 2

Donde *C* es un valor entero empírico mayor que 1, *w* es el vector de coeficientes, *b* es una constante que caracteriza el hiperplano, es el kernel empleado y *£i* es un parámetro que permite manejar los datos de entrada no separables. El índice *i* etiqueta los casos de entrenamiento N. Debe tenerse en cuenta que *y € + 1* es la clase etiquetada y x_i es la variable independiente. El *kernel Φ* se usa para transformar los datos de entrada al espacio de nuevas características. Cabe señalar que mientras mayor es C, mayor es el error penalizado. Por lo tanto, C deber ser escogido con cuidado para evitar el sobre entrenamiento. [7]

1.3 Máquina de Soporte vectorial para Regresión

Las Máquinas de Soporte Vectorial se desarrollaron inicialmente para solucionar problemas de clasificación, pero se han ampliado para problemas de regresión. Los resultados finales a los que se puede arribar luego del empleo de las SVM pueden ser cualitativos o cuantitativos, para en el análisis cuantitativo se emplean Máquinas de Soporte Vectorial para regresión. Dicho método es una extensión del anteriormente explicado donde se incluyen los estimadores de rangos. El empleo de estos determinan los valores que tienen ruido dentro de la predicción a través de funciones de pérdida, donde los primeros pasos en este sentido se dieron por Tuckey quien demostró que en situaciones reales, se desconoce el modelo del ruido y dista de las distribuciones supuestas. A raíz de esto, Huber crea el concepto de estimadores robustos los cuales están determinados por funciones de pérdida. En la actualidad, las más utilizadas son: las funciones de pérdida cuadrática y lineal, y la de Huber, entre otras.

En este tipo de técnicas, su estructura se determina sobre la base del conjunto de entrenamiento necesitándose pocos parámetros para el mismo. Dicho entrenamiento se reduce a la solución de un problema de optimización que se reduce a un problema de programación cuadrática. Al mismo tiempo, el uso de las funciones Kernels muestra una gran eficiencia en el resultado de la predicción. Los tipos de Máquinas de Soporte Vectorial para la regresión, presentes en la investigación son Epsilon_SVR y NU_SVR.

1.5 Kernel

Las funciones *kernel* son funciones matemáticas que se emplean en las Máquinas de Soporte Vectorial. Estas funciones son las que le permiten a las MSV convertir un problema de clasificación no lineal en el espacio dimensional original, a un problema de clasificación lineal en un espacio dimensional mayor. Para

que una función pueda ser considerada candidata a kernel, debe ser continua, simétrica y positiva. Existen diversos *kernels* entre los cuales se destacan el lineal, el RBF (Función de Base Radial), el Polinomial, el Sinusoidal y otros [8, p. 4]. Se recomienda para su empleo el kernel RBF. Esto se puede visualizar en la Figura 5:

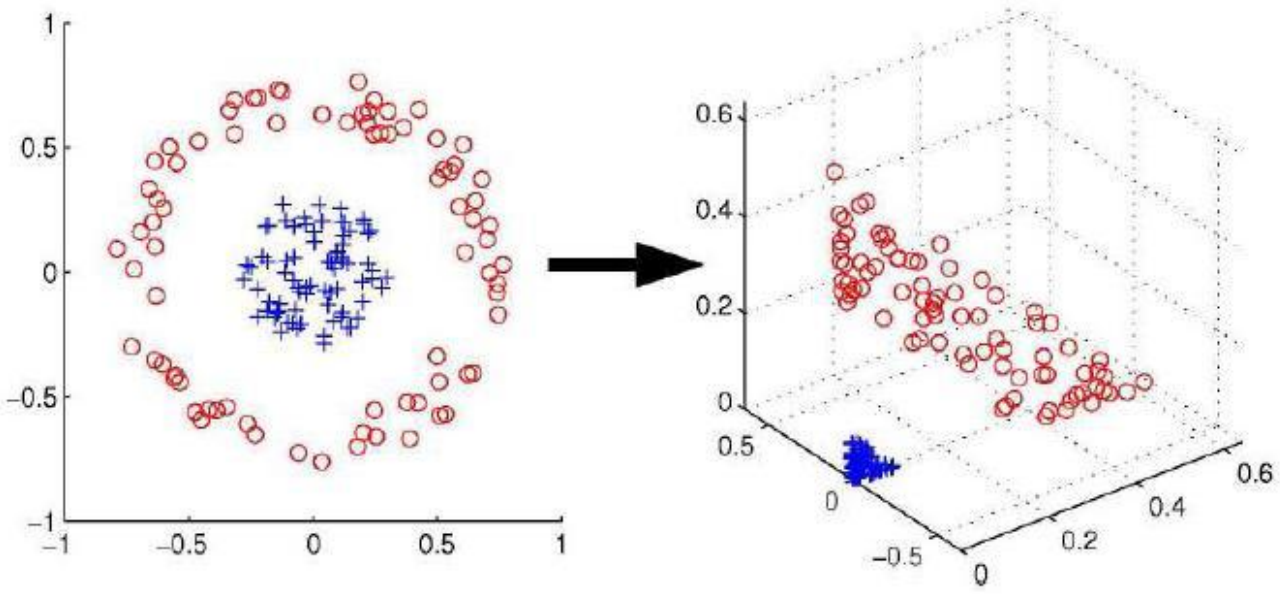

Figura 4. "La transformación de los datos puede hacerlos linealmente separables"

Algunos ejemplos de funciones *kernel* son:

- LINEAR: u'*v
- POLY: (gamma*u'*v + coef0) ^{dgree}
- RBF: exp(-gamma*|u-v|^2)
- SIGMOID: tanh(gamma*u'*v + coef0)

1.6 Estudios realizados en los últimos años

Las Máquinas de Soporte Vectorial han tenido gran aplicación en la bioinformática, principalmente en la predicción de actividad biológica en compuestos y en muchos otros campos de estudio en los últimos años, por ejemplo:

Año 2001

Se emplearon en el diseño de medicamentos para análisis de datos farmacéuticos en la Universidad de Londres. [9]

Año 2003

Fueron utilizadas para el diseño de fármacos y el estudio de similitud de los Medicamentos y predicción de la inhibición de enzimas. En el Departamento de Química de la Universidad de Moscú. [10]

Año 2004

Las Máquinas de Soporte Vectorial fueron utilizadas en la Universidad de París para desarrollar los Modelos QSAR que correlacionan las estructuras moleculares a su toxicidad y bioactividades. El funcionamiento y la capacidad predictiva de Máquinas de Soporte Vectorial, fueron descritas usando los parámetros fisicoquímicos o los descriptores moleculares. En ambos casos estudiados, la capacidad predictiva del modelo de Máquinas de Soporte Vectorial es comparable o superior. Los resultados indican que Máquinas de Soporte Vectorial se puede utilizar como herramienta para modelar los estudios QSAR. [11]

Se empleó en la Universidad de París al efectuar el estudio cuantitativo de la relación estructura-movilidad de los ácidos carboxílicos en la electroforesis de vasos capilares. [12]

Las Máquinas de Soporte Vectorial fueron utilizadas en el Departamento de Química de la Universidad de Lanzhou para desarrollar un modelo cuantitativo de la relación de la Estructura-Característica (QSPR) de la energía en enlace de la disociación del Oxígeno con el hidrógeno (O-H) de 78 fenoles substituidos. [8]

Las Máquinas de Soporte Vectorial, como tipo novedoso de máquina que aprendía, fueron utilizadas para desarrollar un modelo cuantitativo de la relación de la estructura-movilidad (QSMR) de 58 alifáticos y de ácidos carboxílicos aromáticos basados en los descriptores moleculares. [13]

Año 2005

La Universidad de Lanzhou utilizó Máquinas de Soporte Vectorial en el estudio de Modelos QSAR de los compuestos de interrupción de la endocrina natural, sintética y ambiental para atar al receptor del andrógeno. Donde a partir de cinco descriptores el método mostró excelentes resultado. [14]

Año 2006

Fue empleada para determinar la predicción de un modelo cuantitativo de la relación de la estructura-característica (QSPR) se ha desarrollado para la degradación electroquímica de fenoles substituidos. Comparado con los modelos desarrollados con los cuadrados (PLS) y la regresión linear múltiple (MLR), donde estaban 0.804 y 0.799 Q2 respectivamente, Máquinas de Soporte Vectorial demostraron rendimientos más altos.[15]

Se emplearon para predecir relaciones cuantitativas de la estructura característica para los pesticidas en la universidad de Lanzhou, donde se probaron con: método heurístico (HM) y Máquinas de Soporte Vectorial y los modelos lineales y no lineales, generados por ambos se compararon los resultados de estos dos métodos, resultando las Máquinas de Soporte Vectorial el mejor método. [16]

Se emplearon en relación cuantitativa, la estructura-actividad de modelos para la predicción de los irritantes sensoriales de productos químicos orgánicos volátiles. De ellos se han desarrollado: modelos de clasificación y de regresión para la predicción de los irritantes sensoriales (LOGRD50) de los productos químicos orgánicos volátiles (VOCs). Cada compuesto fue representado por los descriptores estructurales. [17]

Año 2007

En un estudio conjunto entre las venezolanas Universidad Nacional Experimental Francisco de Miranda y la Universidad Simón Bolívar y el Real Hospital Victoria y la Universidad de Ulster del Reino Unido se aplicaron las Máquinas de Soporte Vectorial para la predicción de la ocurrencia de Shocks eléctricos en la cardioversión interna de pacientes con fibrilación atrial también conocida como auricular. [18]

En Cuba

El uso de las Máquinas de Soporte Vectorial en nuestro país es insipiente. Algunos ejemplos de la misma se muestran a continuación:

La Universidad Central de las Villas: “Modelo de máquinas de vectores de soporte para regresión aplicado a la estimación de la tensión de ruptura por termofluencia en aceros ferríticos”. [19]

El Centro de Aplicaciones de Tecnologías de Avanzada: “Las Máquinas de Soporte Vectorial para la clasificación y calibración multivariante y multivía en Quimiometría”. [20]

En la Universidad de las Ciencias Informáticas se realizó un ensayo que demostró la aplicabilidad de las Máquinas de Soporte Vectorial en la predicción de actividad biológica en compuestos orgánicos.

1.7 Softwares vinculados a las Máquinas de Soporte Vectorial a nivel mundial

Weka

Entorno para Análisis del Conocimiento de la Universidad de Waikato) es un conocido software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato.

Weka es una colección de algoritmos de aprendizaje automático para la extracción de datos tareas. Los algoritmos pueden ser aplicados directamente a un conjunto de datos o llamar desde su propio código Java. Weka contiene herramientas para los datos de pre-procesamiento, clustering, reglas de asociación, y la visualización, así como clasificación, regresión usando las Máquinas de Soporte Vectorial. También es muy adecuado para el desarrollo de nuevos planes de aprendizaje máquina. [21]

Librería SVM

LIBSVM es un software integrado para la clasificación por vectores de soporte, (C-SVC, nu-SVC), para la regresión (Epsilon-SVR, nu-SVR) y la estimación de la distribución (una clase SVM). Soporta multi-clasificación de clase. [22]

1.8 Selección de Variables

Previo a un problema de clasificación, la selección de variables independientes de un conjunto dado, consiste en encontrar un subconjunto de ellas que resulten relevantes y con un mínimo de redundancia en el contenido de información, para llevar a cabo la tarea de clasificar de forma óptima [23]. El problema de selección de variables para cantidades de variables como las utilizadas en este trabajo, el análisis del espacio de todas las posibles combinaciones demanda mucho tiempo de procesamiento. En este sentido, las diferentes estrategias utilizan para la reducción de dicho espacio los métodos de Efromyson, Furnival y Wilson [24]. Posteriormente, se comienzan a profundizar y mejorar los criterios estadísticos de selección.

Los más aceptados son los propuestos por Mallows [25], Akaike[26] y Schwarz[27], los cuales se basan en la familia de criterios de la suma de cuadrados penalizada. Más recientemente se han propuesto nuevos ajustes basados en esta familia de criterios, entre los cuales figura el *Risk Inflation Criterion* propuesto por Foster y George [28].

La búsqueda de un subconjunto de variables es un problema de tipo No Polinomial completo (NP-completo), los cuales no tienen solución práctica. El uso de meta-heurísticas permite obtener soluciones razonablemente buenas sin explorar todo el espacio de soluciones.

En un sistema de aprendizaje, se pueden utilizar diversas herramientas en combinación con el propio algoritmo de aprendizaje, como por ejemplo: la continuación de características y la sustitución de valores nulos. En la presente tesis, se trabajará con modelos basados en el aprendizaje supervisado [29]. Para los modelos basados en aprendizaje supervisado existen diferentes métodos para la selección de variables.

Los algoritmos de búsqueda más frecuentemente reportados para la realización de estudios de relación entre la estructura química y la actividad biológica, son los Algoritmos Genéticos y el Enfriamiento Simulado. **(Ver anexo 1).**

1.9 Servicios Web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para inter-operar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

1.9.1 Funcionamiento

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

Para optimizar el rendimiento de las aplicaciones basadas en Servicios Web, se han desarrollado tecnologías complementarias a SOAP, que agilizan el envío de los mensajes (MTOM) y los recursos que se transmiten en esos mensajes (SOAP-RRSHB).

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes.

Durante la evolución de las necesidades de las aplicaciones basadas en Servicios Web de las grandes organizaciones, se han desarrollado mecanismos que permiten enriquecer las descripciones de las operaciones que realizan sus servicios mediante anotaciones semánticas y con directivas que definen el comportamiento. Esto permitiría encontrar los Servicios Web que mejor se adapten a los objetivos deseados. Además, ante la complejidad de los procesos de las grandes aplicaciones empresariales, existe una tecnología que permite una definición de estos procesos mediante la composición de varios Servicios Web individuales, lo que se conoce como coreografía. [30]

1.10 Plug-ins

Un **complemento** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como **plug-in** (del inglés "enchufable"), **add-on** (agregado), **complemento**, **conector** o **extensión**.

Los complementos permiten:

- que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones
- reducir el tamaño de la aplicación
- separar el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software

Un ejemplo de aplicaciones que incluyen complementos son los IDEs de programación.

Los IDEs de programación permiten incluir nuevas herramientas para el desarrollo de aplicaciones, incluso para la programación en diferentes lenguajes de programación. Dos ejemplos muy conocidos son el

Eclipse y el **NetBeans**. En sentido general, cualquier aplicación puede añadir soporte para complementos. El primer complemento se diseñó en 1987 para el programa HyperCard de Macintosh.

1.10.1 Front-End GRATO

El Front-End GRATO surgió como una necesidad del Proyecto alasGRATO en el Polo de Bioinformática de la Facultad 6 de la Universidad de las Ciencias Informáticas para mejorar la interacción de los usuarios con la plataforma. Esta herramienta soluciona la principal dificultad que tenían las otras herramientas manejadoras de plug-ins antes mencionadas, o sea, la curva de aprendizaje lenta que presentan estas.

El Front-End GRATO es dinámico y multiplataforma, y se encarga de la portabilidad de los plug-ins garantizando:

- Unificar la carga de los plug-ins a través de descriptores XML.
- Ganar en soportes de escalabilidad para el ingreso de futuros plug-ins.
- Ganar en el manejo de memoria en ejecución controlando todos los componentes visuales desde un mismo marco.
- Mantener el principio de conservación, pues en la evolución del Front-End, los plug-ins desarrollados siempre serán compatibles. [31]

Debido a todo lo planteado se escogió al Front-End GRATO como la herramienta a utilizar para el desarrollo de los plug-ins, y así integrar el módulo de predicción de actividad biológica por Máquinas de Soporte Vectorial a la plataforma alasGRATO. Teniendo en cuenta también que este fue creado específicamente para la plataforma alasGRATO con el objetivo de incluirle en forma de plug-in todos los módulos existentes en la misma y que su uso es mucho más sencillo ya que posee una interfaz amigable y configurable.

1.11 Metodología y herramientas

La propuesta de herramientas, tecnologías y metodología que se muestra a continuación surge inicialmente a partir de haber determinado modificar el módulo de predicción de actividad biológica del proyecto alasGRATO para dar cumplimiento al objetivo propuesto. Por sus características, la aplicación es de escritorio, está programada en Java utilizando Eclipse como entorno de desarrollo, el servidor web que utiliza es Apache Tomcat y la base de datos esta creada en MySQL. Todas son herramientas y

tecnologías libres, de ahí que sean las propuestas para el desarrollo de la.

1.11.1 Metodología (Open UP)

OpenUP es un proceso unificado de desarrollo que aplica acercamientos iterativos e incrementales dentro de un ciclo de vida estructurado. Además abraza una filosofía pragmática, ágil, y enfocada en la naturaleza de colaboración del desarrollo del software. Es una herramienta que se puede ampliar a una gran variedad de tipos de proyecto.

El esfuerzo personal en un proyecto de OpenUP se organiza en micro-incrementos, los cuales representan unidades cortas de trabajo que producen un paso constante y mensurable en el progreso del proyecto. El proceso aplica la colaboración intensiva como sistema de desarrollo incremental. Estos micro-incrementos proporcionan un lazo de regeneración extremadamente corto, que conducen a decisiones adaptables dentro de cada iteración. El ciclo de vida de un proyecto en OpenUP está estructurado en cuatro fases: inicio, elaboración, construcción y transición. [32]

1.11.2 Lenguaje de modelado (UML)

El Lenguaje de Modelado Unificado (UML - Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos. Este lenguaje fue creado por un grupo de estudiosos de la Ingeniería de Software formado por: Ivar Jacobson, Grady Booch y James Rumbaugh en el año 1995. Desde entonces, se ha convertido en el estándar internacional para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación orientada a objetos.

UML no es un lenguaje de programación, sino un lenguaje de propósito general para el modelado orientado a objetos. También puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

1.11.3 Herramientas CASE (Visual Paradigm)

Como herramienta CASE(Computer Aided Software Engineering) se empleó Visual Paradigm para el trabajo con UML. La herramienta está diseñada para una amplia gama de usuarios que incluye a: ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas,

interesados en la creación de grandes sistemas de software de manera confiable, a través del paradigma orientado a objetos. VP-UML soporta los últimos estándares de anotaciones de JAVA y UML y provee soporte para la generación de código y la ingeniería inversa para Java. Además, VP-UML se integra con Eclipse, Borland® JBuilder®, NetBeans IDE/Sun™ ONE, IntelliJ IDEA™, Oracle JDeveloper y BEA WebLogic Workshop™, para soportar las fases de implementación en el desarrollo de software. Las transiciones del análisis al diseño, y de éste a la implementación, están adecuadamente integradas dentro de la herramienta CASE, de manera que reduce significativamente los esfuerzos de todas las etapas del ciclo de desarrollo de software.

1.11.4 Lenguaje de programación (Java)

El lenguaje de programación Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos. Su diseño fue concebido para que los programadores puedan lograr fluidez con el lenguaje. Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello, Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje, además, el mismo elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (recolector de memoria dinámica o de basura). No es necesario preocuparse de liberar memoria, el recolector se encarga de ello y como es un thread (hilo) de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de esta.

Una de las características más importante de Java es que posee una arquitectura neutral, es decir el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

Actualmente existen maquinas virtuales de java para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando para la portabilidad a otras plataformas.

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Java, además, construye sus interfaces de usuario a través de un sistema abstracto de ventanas, de forma que ellas puedan ser implantadas en entornos Unix, Pc o Mac.

1.11.5 Entorno de desarrollo (Eclipse)

Eclipse es uno de los más potentes entornos integrados de desarrollo (IDE) que permite la construcción de aplicaciones en Java. Admite la incorporación de otros plug-ins para obtener un mayor número de funcionalidades, es una herramienta de código abierto, multiplataforma y compatible con los sistemas operativos más utilizados en el mundo. Además posee la capacidad de ser soportado para distintas arquitecturas, control de versiones con cvs o con subversión, resaltado de sintaxis, un potente refactorizado de código, asistentes (wizards): para la creación, exportación e importación de proyectos; permite la integración con la herramienta CASE Visual Paradigm.

1.11.6 Gestor de Base de Datos (MySQL)

Como sistema gestor de base de datos se escogió MySQL en su versión 5.0 ya que es uno de los gestores más rápidos que se encuentran en el mercado. Presenta versiones en varios sistemas operativos y compatibilidad entre sus versiones y es muy fácil de usar. MySQL es un sistema fácil de instalar y configurar en servidores Windows, Linux entre otros. Compite con sistemas RDBMS como Oracle, *SQL Server* entre otros. Incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuarios, administrar el sistema y proteger los datos. Permite centralizar la base de datos y brinda mayor seguridad para estas. Dispone de muchas funciones vitales para el desarrollo profesional cómo puede ser el volcado online, la duplicación etc.

SQLite

SQLite usa un sistema de tipos inusual, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, durabilidad y la mayor parte de las consultas complejas. El motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en

el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

Conclusiones del Capítulo

En este capítulo se hizo un estudio sobre métodos que se utilizan para predecir actividad biológica y se estudiaron algunos softwares que permiten este tipo de estudios. Se mencionan los tipos de máquinas de soporte Vectorial y se describen las MSV para regresión y clasificación. Se seleccionó como herramienta manejadora de plug-ins al front-end alasGRATO para la integración de los módulos de predicción de actividad biológica con las Máquinas de Soporte Vectorial, como metodología de desarrollo de software OpenUP, como lenguaje de modelado UML y la herramienta CASE escogida para el trabajo con UML es Visual Paradigm, como lenguaje de programación Java y la herramienta de desarrollo, el Eclipse. Como gestor de base de datos tenemos MySQL y SQLite.

CAPíTULO 2 CARACTERíSTICAS DEL SISTEMA

Introducción

En este módulo para predicción de la actividad biológica para la plataforma alasGRATO, el proceso de negocio definido es muy simple lo que trae consigo la realización de un modelo de dominio para identificar los procesos de automatización: realizar selección de variables, realizar predicción y realizar entrenamiento. Teniendo como actores principales de nuestro análisis al Especialista Químico para lograr una predicción de actividad biológica anticancerígena a partir de fragmentos moleculares, empleando MSV como técnica de inteligencia artificial.

2.1 Modelo de Dominio

El flujo de trabajo de Modelamiento del Negocio describe los procesos del negocio, identificando quiénes participan y las actividades que requieren automatización. Teniendo como objetivo:

- Comprender la estructura y la dinámica de la organización en la cual se va a implementar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

Como parte del Modelo de Negocio se encuentra el Modelo de Dominio el cual captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en que trabaja el sistema. Los conceptos definidos para dicho modelo son:

- **Especialista Químico:** Cliente que utilizará en sistema.
- **Entrenamiento:** Acción que se realiza para el aprendizaje de la máquina de soporte vectorial.
- **Modelo:** Modelo de máquina de soporte vectorial que se crea para realizar una predicción.
- **Predicción:** Acción que el sistema realiza para emitir un resultado.

- **Selección de Variables:** Reducción del número de variables.
- **Descriptores:** Valores matemáticos que caracterizan estructuralmente la molécula.
- **Actividad Biológica:** Actividad asociada a una molécula.

2.1.1 Descripción del modelo de dominio

Como se representa en el modelo de dominio todas las acciones, son realizadas por el Especialista Químico. La reducción de variable se realiza para eliminar las variables de los descriptores de entrada que no aporten suficiente información. El entrenamiento es realizado para la generación de los modelos ya sean de clasificación o regresión, este entrenamiento requiere de descriptores moleculares y la actividad biológica anticancerígena asociada. Los modelos generados son los que se usarán luego para la predicción, la cual requiere además datos de pruebas que no son más que descriptores moleculares.

Modelo de Dominio

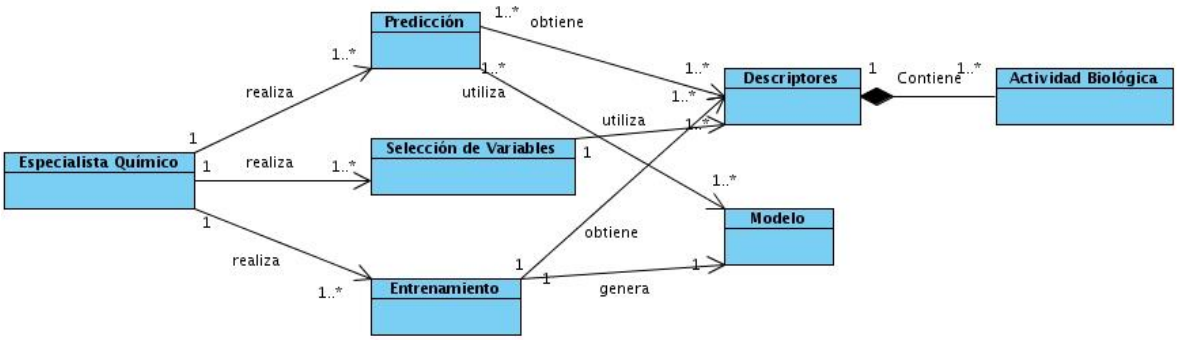

Figura 5. Modelo del Dominio del Sistema

2.2 Requerimientos del Sistema

2.2.1 Requisitos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Para el sistema propuesto se definen:

RF1.- Cargar descriptores moleculares.

RF2.- Realizar Selección de Variables.
RF3.- Seleccionar medida de evaluación.
RF4.- Seleccionar método de búsqueda.
RF5.- Seleccionar MSV de Clasificación.
RF6.- Seleccionar MSV de Regresión.
RF7.- Seleccionar Ensayo de la BD.
RF8.- Generar Modelos de Clasificación.
RF9.- Generar Modelos de Regresión.
RF10.- Cargar Modelo de la BD.
RF11.- Cargar Modelo de la BD local.
RF12.- Cargar Modelo del Fichero.
RF13.- Realizar Predicción por Clasificación.
RF14.- Realizar Predicción por Regresión.
RF15.- Guardar Fichero de Predicción.
RF16.- Eliminar Modelo.

2.2.2 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Representan las características del producto.

Apariencia o interfaz externa

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

Usabilidad

- El sistema podrá ser usado por cualquier tipo de persona que posea conocimientos básicos en el manejo de la computadora. En este caso el especialista químico.

- Se necesita que el usuario sea un especialista en química para entender los resultados dados por la aplicación.

Rendimiento

- Al estar concebida para un ambiente cliente/servidor, se trata de garantizar la rapidez de respuesta del sistema ante las solicitudes de los usuarios, al igual que la velocidad de procesamiento de la información. Para lo cual se realiza la validación de los datos y la manipulación de eventos en el cliente y en el servidor aquellas que por cuestiones de seguridad, o de acceso a los datos lo requieran. Lográndose así un tiempo de respuesta más rápido, una mayor velocidad de procesamiento, y un mayor aprovechamiento de los recursos.

Soporte

- El sistema debe propiciar su mejoramiento y la incorporación de otras opciones en un futuro.

Portabilidad

- El sistema puede ser ejecutado sobre los sistemas operativos Linux y Windows, por su característica de ser multiplataforma.

Seguridad

- El especialista solo tiene control a predecir y de realizar el modelo a través del entrenamiento.

Confiabilidad

- El sistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error.

Ayuda

- La aplicación posee ayuda, en la que se explica de forma clara el uso de las opciones del sistema, garantizando así el buen desempeño de los usuarios a la hora de interactuar con el mismo.

Software

- Se debe disponer de sistemas operativos *Linux, Windows* 95 o superior para la instalación de la aplicación.
- Debe tenerse instalado el *Java Runtime Environment* (JRE) versión 1.6 o superior.
- Un servidor Apache-Tomcat como servidor de aplicaciones.

Hardware

- Para el desarrollo y puesta en práctica del proyecto se requieren máquinas con los siguientes requisitos:

  Máquinas Clientes:

  - Procesador *Pentium* 3 o superior.
  - Mínimo 256 Mb de RAM.
  - Mínimo 50 Mb de capacidad del disco duro.

  Máquina servidor:

  - Procesador Pentium 4 o superior.
  - Mínimo 512 Mb de RAM.
  - Mínimo 2 GB de capacidad de disco duro.

2.3 Definición del Sistema

2.3.1 Actores

Se le llama actor a toda entidad externa al sistema que guarda una relación con este y que le demanda una funcionalidad. Esto incluye a los operadores humanos pero también incluye a todos los sistemas externos así como a entidades abstractas como el tiempo. Es decir el personal que trabaja directamente con la aplicación en este contexto sería el Especialista Químico el actor definido.

Actores	Descripción

Especialista Químico	Usuario que hace uso del sistema, para obtener la predicción de algún compuesto químico.

Tabla 1. Descripción de los Actores del Sistema

2.3.2 Casos de Uso del Sistema

Los casos de uso del sistema son la funcionalidad que el mismo tendrá, estarán estrechamente relacionados con los requisitos funcionales planteados anteriormente.

2.3.3 Diagrama de Casos de Uso del Sistema

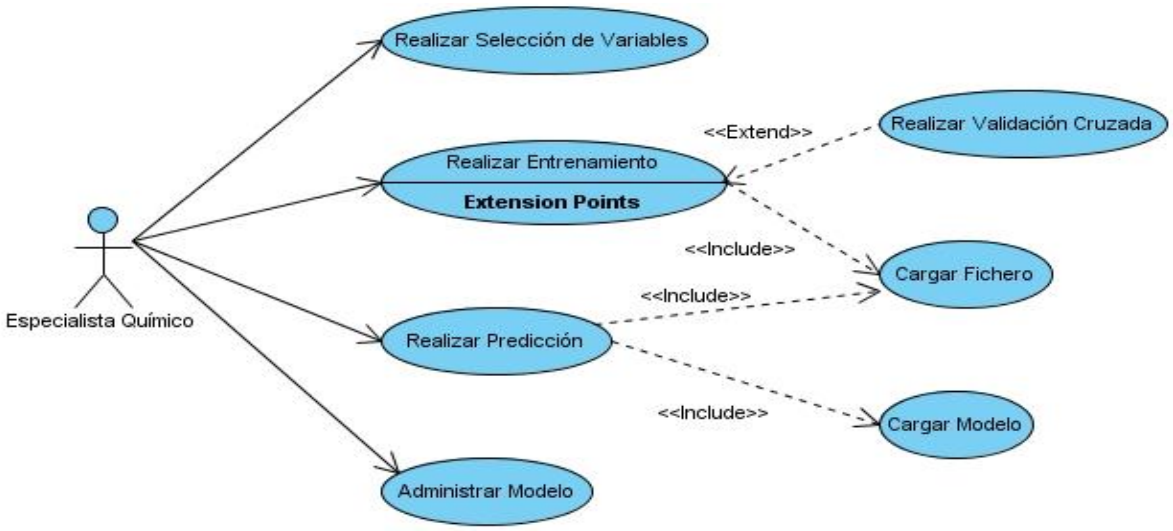

Figura 7. Diagrama de Casos de Uso del Sistema

2.3.4 Descripción de los Casos de Uso del Sistema

- **Descripción del CUS:** Realizar Selección de Variables.

Caso de Uso:	Realizar Selección de Variables
Actores:	Especialista químico
Propósito:	Reducir las variables de los datos, para lograr una mejor predicción.
Resumen:	El Especialista químico inicia el caso de uso cargando los

<table>
<tr><td></td><td colspan="2">ficheros arff que le desea reducir las variables. Selecciona los métodos de evaluación y los métodos de búsqueda, el sistema guarda los ficheros reducidos para ser usados luego en entrenamiento o predicción.</td></tr>
<tr><td>Referencia:</td><td colspan="2">RF1, RF2, RF3, RF4</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">Se deben encontrar los datos a los que se les van a realizar la reducción de variable.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona la opción Cargar Fichero.</td><td>2. El sistema muestra una ventana para especificar el lugar de dónde se cargará el fichero.</td></tr>
<tr><td colspan="2">3. El Especialista químico selecciona el fichero que quiere cargar y presiona abrir.</td><td>4. El sistema verifica que el fichero a cargar sea .arff.
5. El sistema carga el fichero en el árbol de ficheros.</td></tr>
<tr><td colspan="2">6. El Especialista químico selecciona la(s) molécula(s) que le quiere realizar la reducción.</td><td>7. El sistema muestra la representación según la selección realizada por el especialista.</td></tr>
<tr><td colspan="2">8. El Especialista químico escoge los parámetros a tener en cuenta :
• Métodos de búsqueda:
  • Híbrid
  • Genéric
  • Simulated anealing
• Medidas de evaluación:
  • ConsistencySubsetS
  • CF_sSubset
  • CF_sLocalPrediction
• Cardinalidad
• Probabilidad de cruzamiento
• Salida múltiple</td><td>9. El sistema muestra la representación según la selección realizada por el usuario.</td></tr>
</table>

• Probabilidad de mutación • Generación máxima • Tamaño de la población	
10. El Especialista químico selecciona el botón realizar la reducción.	11. El sistema verifica que exista al menos una molécula seleccionada. 12. El sistema verifica que los datos introducidos son correctos y/o que los campos obligatorios no están vacíos. 13. El sistema reduce las variables de los datos cargados. 14. El sistema crea el fichero con los datos reducidos. Terminando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.a El sistema muestra un mensaje de error ¨Error de formato en el fichero¨ en caso de que no tenga uno de los dos formatos y se retorna al flujo normal de eventos número 3.
3.a El Especialista químico selecciona el botón Cancelar.	4.b El sistema cierra la ventana abierta.
	11.a El sistema muestra un mensaje de error ¨¨No se seleccionó ninguna molécula¨ y se retorna al flujo normal de eventos número 6.
Poscondiciones	El sistema escribe en un fichero los datos reducidos. El sistema guarda el fichero en la base de datos.
Prioridad:	Crítico.

Tabla 2. Descripción del Caso de Uso Realizar Selección de Variables

- **Descripción del CUS:** Realizar Predicción.

Caso de Uso:	Realizar Predicción
Actores:	Especialista químico

Propósito:	Realizar la predicción de los fragmentos moleculares y devolver dicha predicción.
Resumen:	El Especialista químico inicia el caso de uso al cargar el fichero de prueba con la molécula o las moléculas de la cual desea saber su predicción, escoge un modelo que corresponda al entrenamiento realizado a ese tipo de datos y asigna la dirección de un archivo vacío para la escritura de los resultados.
Referencia:	Caso de Uso Cargar Fichero<incluido>, Caso de uso Cargar Modelo<incluido>, RF7, RF12, RF13, RF14, RF15,
Precondiciones:	Debe estar cargado un fichero con los modelos de entrenamiento, además de existir uno con los datos de prueba y por último uno para escribir los resultados.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista químico selecciona la molécula que le quiere realizar la predicción.	2. El sistema muestra la representación según la selección realizada por el especialista.
3. El Especialista químico especifica la probabilidad.	4. El sistema muestra la representación según el número introducido por el especialista.
5. El Especialista químico selecciona el fichero modelo que corresponda con sus datos de prueba.	6. El sistema brinda la opción al especialista de: • Cargar el modelo de la BD local. • Cargar el modelo de la BD de la plataforma. (Ir al Caso de Uso Cargar Modelo)
7. El Especialista químico selecciona el botón para realizar la predicción.	8. El sistema realiza la predicción con el fichero de prueba y el fichero modelo. 9. El sistema escribe los resultados en el fichero de predicción. 10. El sistema muestra al especialista los

	valores resultantes de la predicción.
11. El Especialista químico selecciona el directorio en el cual se guardará el fichero de la predicción.	12. El sistema guarda la dirección del fichero. Terminando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones:	El sistema crea un fichero con los datos de la predicción. El sistema muestra el resultado de la predicción.
Prioridad:	Crítico.

Tabla 3. Descripción del Caso de Uso Realizar Predicción

- **Descripción del CUS:** Realizar Entrenamiento.

Caso de Uso	Realizar Entrenamiento.
Actores:	Especialista químico
Propósito:	Crear modelos para la predicción a partir del entrenamiento de los datos.
Resumen:	El Especialista químico inicia el caso de uso cuando selecciona las opciones para realizar el entrenamiento, carga el fichero con los datos de los fragmentos que presentan las actividades biológicas asociadas, y el índice del estado refractopológico total, el sistema realiza el entrenamiento y devuelve el modelo.
Referencia:	Caso de Uso Cargar Fichero <incluido>, RF5, RF6, RF8, RF9, Caso de Uso Realizar Validación Cruzada <extendido>
Precondiciones:	Se deben cargar los datos para el entrenamiento desde la Base de Datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Especialista químico selecciona la	2. El sistema muestra la representación

molécula que le quiere realizar el entrenamiento.	según la selección realizada por el especialista.

3. El Especialista químico selecciona los parámetros para el entrenamiento.
• MSV:
  • C-SVC(C)
  • nu-SVC(C)
  • one-Class(C)
• Kernel:
  • lineal
  • polynomial
  • RBF
  • Sigmoid
• Costo
• Degree
• Epsilon(p)
• Epsilon(e)
• Gamma
• nu
• Coeficiente
• Cache size
• Weight
	4. El sistema muestra la representación según la selección realizada por el usuario.

5. El especialista químico selecciona el botón reducir para realizar el entrenamiento.	6. El sistema verifica que los datos introducidos no son correctos y/o que existen campos obligatorios vacíos. 7. El sistema realiza el entrenamiento de los fragmentos. 8. El sistema brinda la posibilidad al especialista de guardar el modelo.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

Precondiciones:	El sistema genera un modelo.
Prioridad:	Crítico.

Tabla 4. Descripción del Caso de Uso Realizar Entrenamiento

Conclusiones del Capítulo

En este capítulo se identificaron los requerimientos funcionales, los actores y casos de uso del sistema que sirvieron de base para realizar el diseño del producto que se presenta en este trabajo. Se mostraron las relaciones entre actores - casos de uso, así como los casos de uso críticos para la arquitectura.

CAPíTULO 3 DISEÑO DEL SISTEMA

Introducción

En el flujo de trabajo de Análisis y Diseño se modela el sistema y se le da la forma mediante la arquitectura, de manera tal que soporte los requisitos funcionales y no funcionales, creando una entrada apropiada y un punto de partida para las actividades de implementación, dando la posibilidad de obtener un producto con calidad y que satisfaga las necesidades del cliente. En el presente capítulo se dará una descripción del estilo arquitectónico utilizado para el desarrollo del sistema, se describirán los patrones de diseño empleados y los diagramas de clases de diseño e interacción de los casos de uso principales. Se mostrará el diagrama de despliegue del sistema.

3.1 Descripción de la Arquitectura

De acuerdo con la IEEE la arquitectura es:

- El nivel conceptual más alto de un sistema en su ambiente.
- La organización fundamental de un sistema descrita en:
  - Sus componentes.
  - Relación entre ellos y con el ambiente.
  - Principios que guían su diseño y evolución. [33]

3.1.1 Patrón arquitectónico

Los patrones arquitecturales son aquellos que expresan un esquema organizativo estructural fundamental para sistemas software (no son otra cosa que los estilos). Un estilo de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico y presenta un

esquema genérico y probado de su solución. La aplicación ha sido desarrollada sido desarrollada bajo la utilización del patrón arquitectónico Modelo Vista Controlador. [34]

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web. Para aplicaciones J2EE, la arquitectura MVC satisface las necesidades de la aplicación. El patrón indica que se deben establecer tres componentes o capas en la arquitectura, y que cada una de estas, solo se comunica con la adyacente.

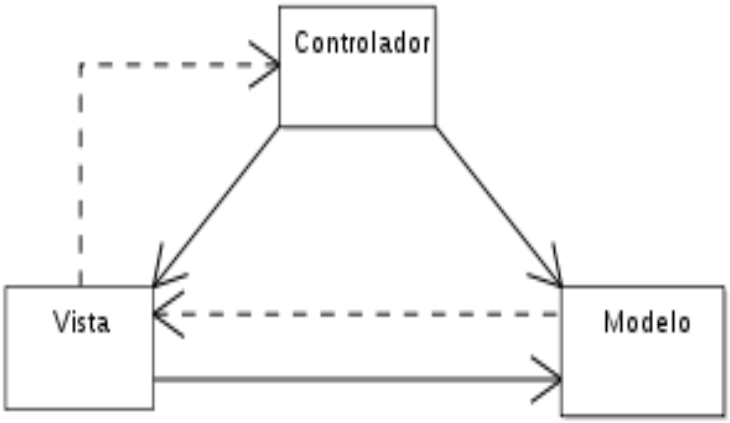

Figura 8. Patrón Modelo Vista Controlador

Descripción de los componentes:

- **Modelo:** Es la representación específica de la información con la cual el sistema opera. Básicamente contiene la lógica de negocio real, el dominio de la aplicación con sus clases, los métodos get y set y los objetos de acceso a datos (DAO). Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

- **Vista:** Presenta el modelo en un formato adecuado para interactuar con el usuario. Responsable de la lógica de presentación y captura de datos de nuestro sistema. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Reciben las entradas, como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio (requests) para el modelo o la vista, traslada las peticiones de la capa Vista a la capa Modelo, y según la respuesta, la reenvía o no a la capa Vista. Carga objetos y opera con ellos.

El uso de este patrón trae asociado directamente un grupo de patrones de diseño como son el patrón Bajo Acoplamiento y el patrón Alta Cohesión.

- **Ventajas que brinda:**

  - Aporta una construcción de software muy mantenible, en la que se pueden localizar de forma ágil los errores.
  - Supone un diseño modular y muy poco acoplado, favoreciendo la reutilización.

3.1.2 Patrones de Diseño

El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuando aplicarlo y si tiene sentido hacerlo teniendo otras restricciones de diseño, como las consecuencias, ventajas e inconvenientes de su uso. Como normalmente tendremos que implementar nuestros diseños, un patrón también proporciona código de ejemplo para ilustrar una implementación. Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final.

J2EE

Los **patrones J2EE** están orientados específicamente a los problemas comunes a todas las aplicaciones J2EE. Algunos están basados en los patrones originales mientras que otros son más específicos del tipo

de problemas que surgen específicamente en J2EE, sea por los tipos de aplicaciones que se suelen desarrollar con la plataforma o por las características (o deficiencias, incluso) de la tecnología. [35]

Patrón Singleton (Instancia única)

Hay muchos casos en los que solo necesitamos una instancia de una determinada clase. Ejemplos típicos son clases que representan las preferencias del usuario o la configuración del sistema, o clases que sirven de interfaz con dispositivos físicos. En estos casos hay que asegurarse de poder obtener una referencia a dicha instancia desde cualquier punto del código, y que solo haya una instancia creada, para evitar posibles problemas o inconsistencias. Una posible solución es definir variables globales (o sea, static). El patrón Singleton nos permite asegurar que de una clase habrá solo una instancia y proporciona un punto de acceso a ella global a todo el código. El diagrama de clases es muy sencillo, ya que se compone de una única clase:

Figura 9. Patrón Singleton

Patrón Façade (Fachada)

Un *Facade* (fachada) consiste en **implementar un interfaz simplificado para un sistema complejo**. La idea es implementar una clase con un interfaz sencillo y que encapsule los detalles de la interacción entre todas las clases del sistema. Es importante notar que se sigue permitiendo el acceso directo a las clases del sistema, a clientes que necesiten "acceso a bajo nivel" pero se simplifica la interacción para los que no necesiten más que operaciones comunes.

Patrón Factory

Debemos crear diferentes objetos, todos pertenecientes a la misma familia. Por ejemplo: las librerías para crear interfaces gráficas suelen utilizar este patrón y cada familia sería un sistema operativo distinto. Así pues, el usuario declara un Botón, pero de forma más interna lo que está creando es un BotónWindows o un BotónLinux, por ejemplo. El problema que intenta solucionar este patrón es el de crear diferentes familias de objetos.

Patrones GRASP(General Responsibility Assignment Software Patterns)

Controlador: El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que: Tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades.

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Alta cohesión: Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización.

3.2 Modelo de Diseño

3.2.1 Diagrama de clases del diseño

Los diagramas de clases se realizan por cada caso de uso, muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras. A continuación se muestran los diagramas de clases del diseño que fueron confeccionados, se han agrupado las clases de acuerdo a su funcionalidad en tres grandes paquetes, haciendo además referencia al patrón arquitectónico expuesto anteriormente: MVC. Se muestran entonces los elementos que componen cada uno de estos paquetes y posteriormente se ilustran los diagramas de clases del diseño realizados para los casos de uso principales.

3.2.3 Diagramas de clases del Diseño para el plug-in de Máquina de Soporte Vectorial

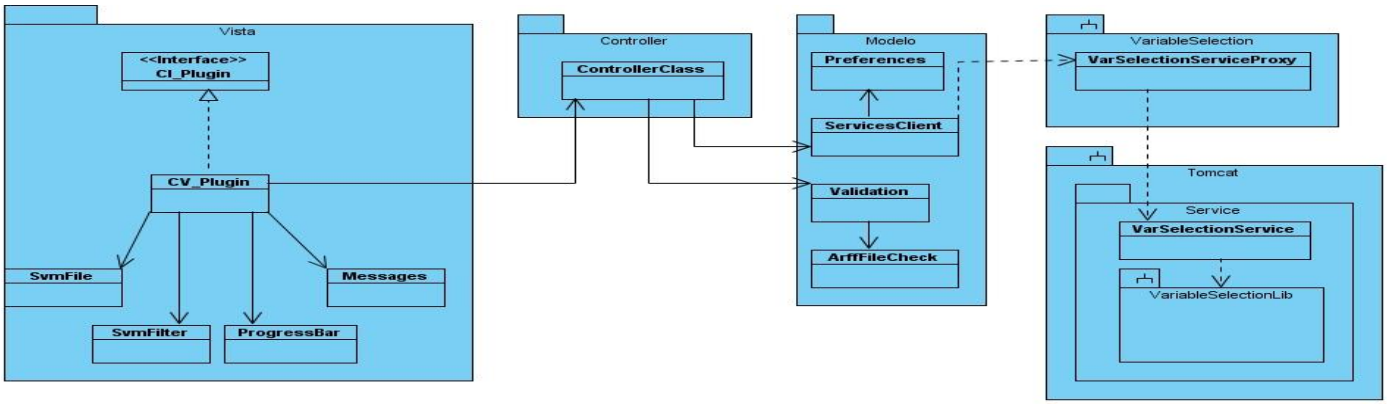

Figura 12. Diagrama de Clases del Diseño del CU Realizar Selección de Variables

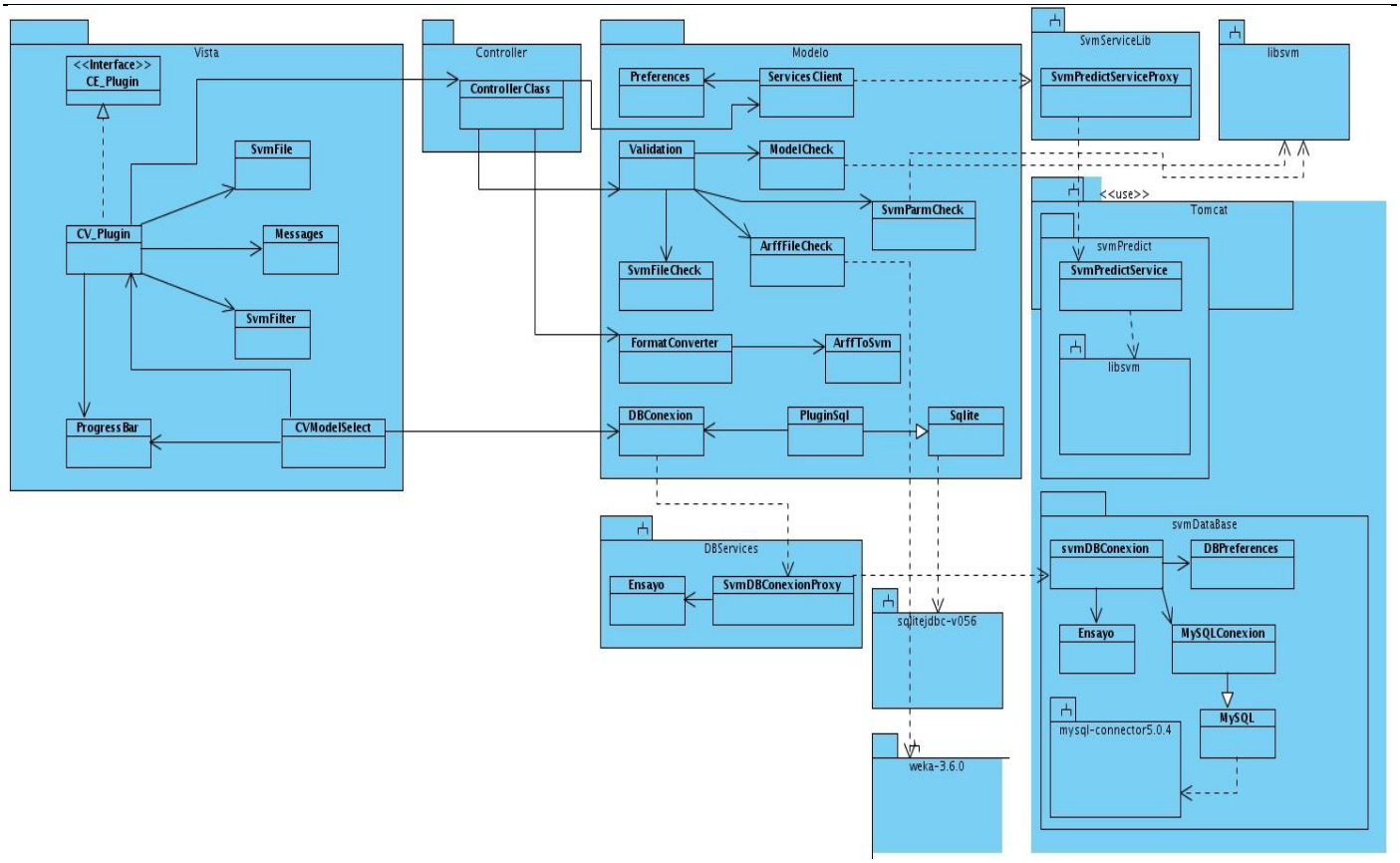

Figura 13. Diagrama de Clases del Diseño del CU Realizar Predicción

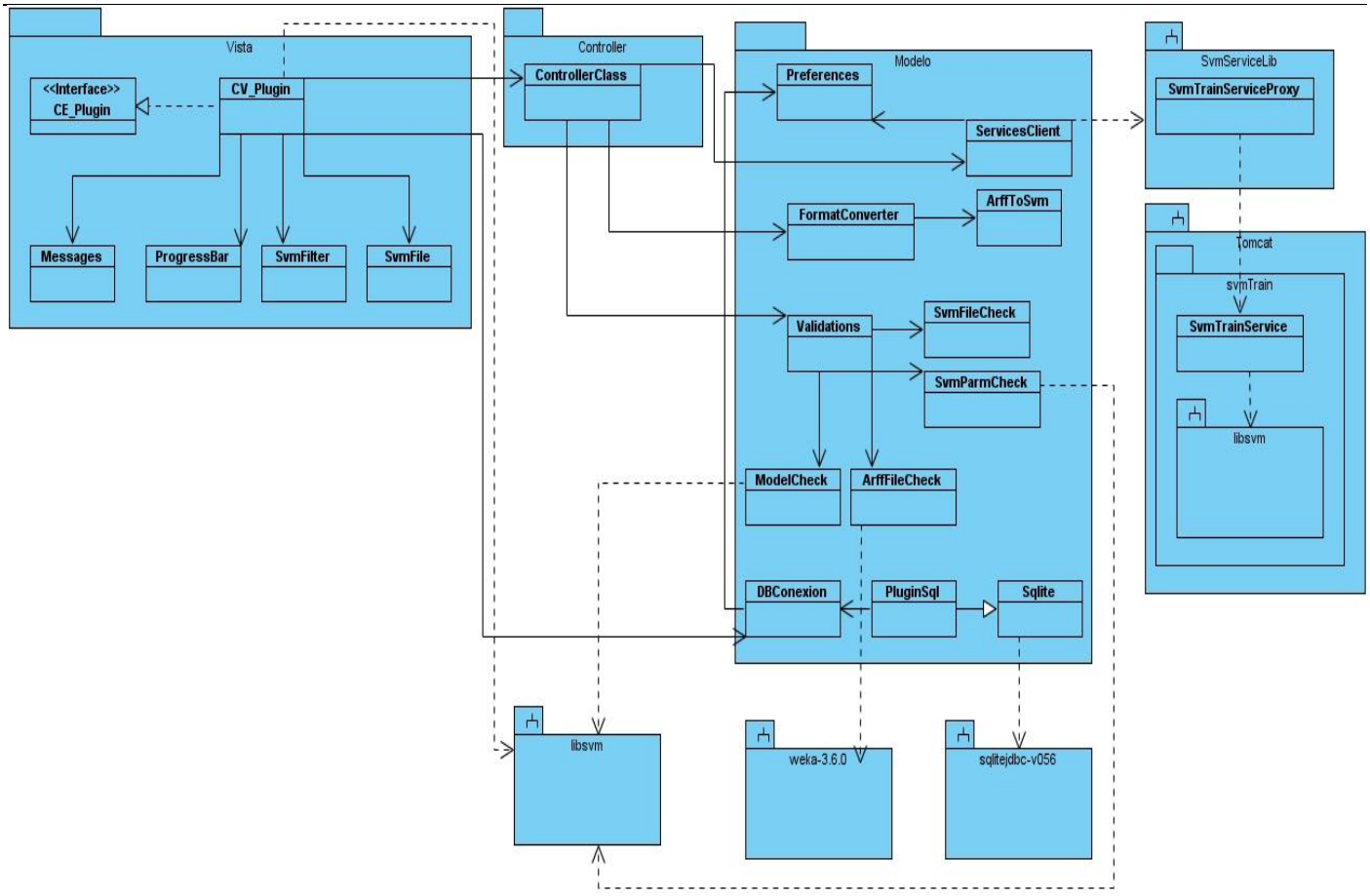

Figura 14. Diagrama de Clases del Diseño del CU Realizar Entrenamiento

Para un mejor entendimiento de los diagramas de clases se efectuó la descripción de las clases. **(Ver planilla: Descripción de las Clases del Diseño)**

3.2.2 Diagramas de interacción (secuencia)

Los diagramas de interacción describen secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Son utilizados para el modelado de los aspectos dinámicos de un sistema, proporcionan una vista integral de su comportamiento. Esto conlleva modelar

instancias concretas, componentes y nodos junto con los mensajes enviados entre ellos que representan su interacción.

Los diagramas de interacción tienen dos formas de manifestarse:

- Diagramas de secuencia.
- Diagramas de colaboración.

Se puede desarrollar o bien el diagrama de secuencia o el diagrama de colaboración, en el trabajo se utilizará el diagrama de secuencia el cual destaca la ordenación temporal de los mensajes, ilustra los objetos que se encuentran en un escenario y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad descrita. Se muestran los diagramas de secuencia para los principales casos de uso.

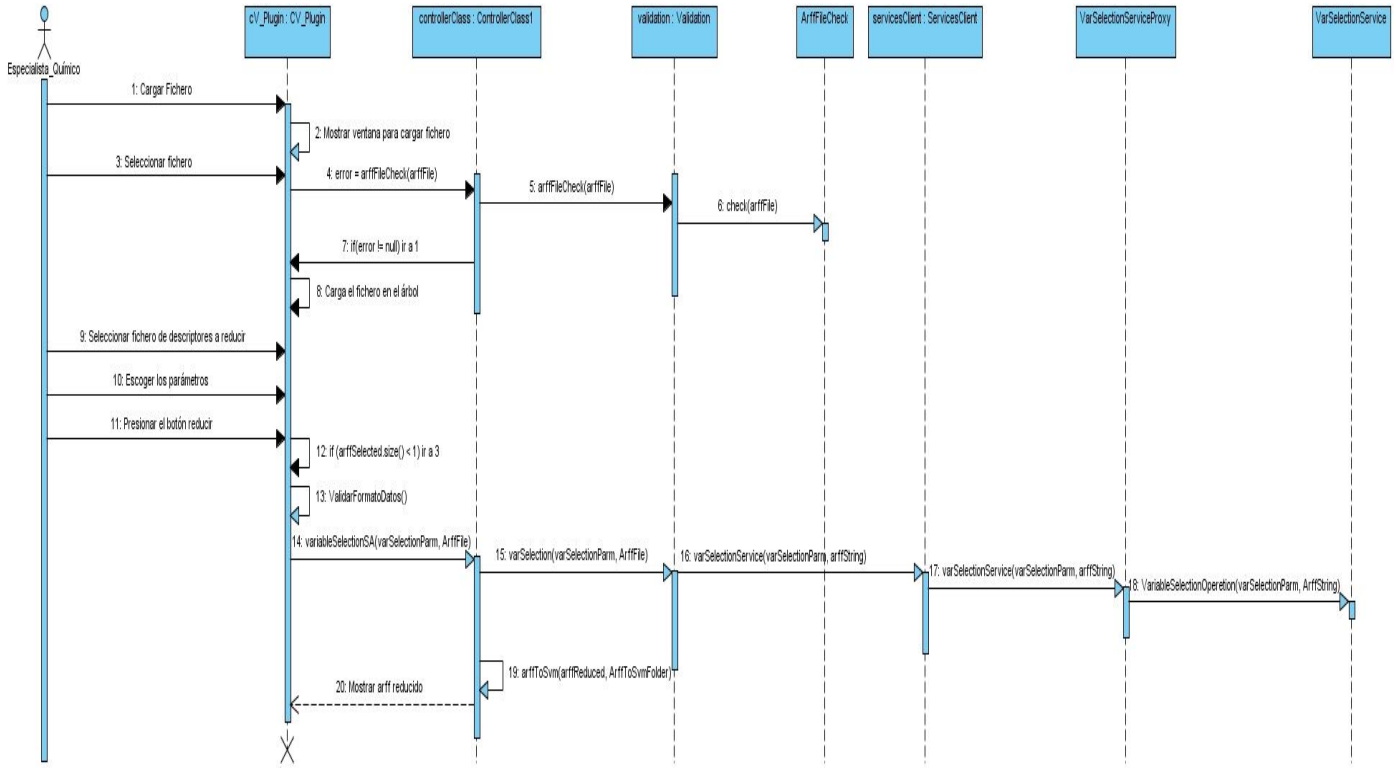

Figura 15. Diagrama de Secuencia del CU Realizar Selección de Variables

Figura 16. Diagrama de Secuencia del CU Realizar Predicción

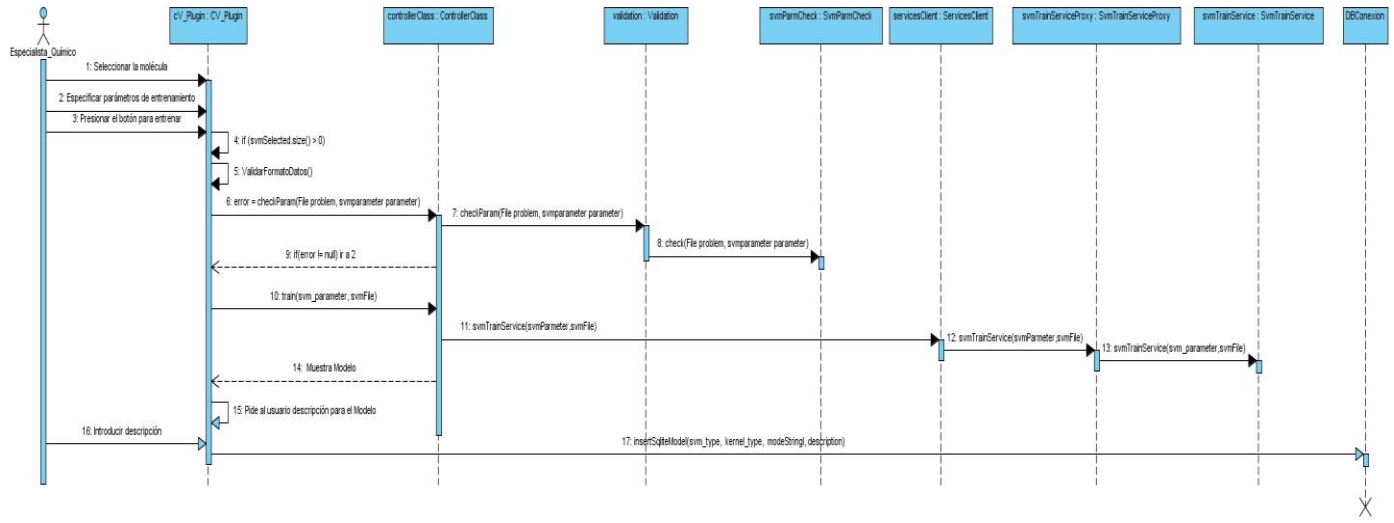

Figura 17. Diagrama de Secuencia del CU Realizar Entrenamiento

3.3 Modelo de Despliegue

El modelo de despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto. A continuación se muestra el diagrama de despliegue para la plataforma alasGRATO.

3.3.1 Diagrama de Despliegue

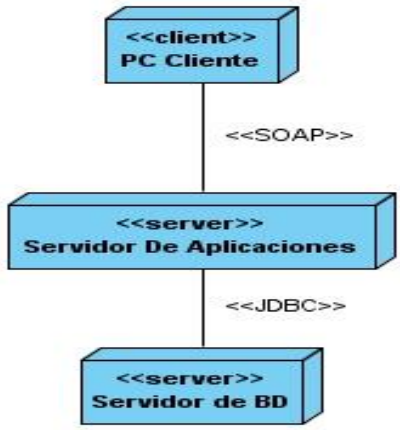

Figura 18. Diagrama de Despliegue

Descripción de los nodos

- **Nodo PC Cliente**

En el Nodo cliente estará el componente predicción de actividad biológica, ya que con este el usuario puede trabajar desde su puesto de trabajo sin hacer ninguna petición al servidor si no lo desea; además se encontrarán los plug-ins que brindarán la interfaz gráfica y la comunicación con los servicios Web.

- **Nodo Servidor de Aplicaciones**

En el nodo servidor estará instalado el servidor de aplicaciones Apache Tomcat, donde se alojarán los servicios Web, que tendrá la plataforma. En este nodo se encontrará, además, un componente de suma importancia como es el Servidor alasGRATO con todos sus plug-ins.

- **Nodo Servidor Base de Datos**

En el nodo servidor de base de datos, se encontrará la base de datos de la plataforma. A esta base de datos se accederá a través del componente acceso a datos DAO que se encontrará en el nodo servidor.

Descripción de la comunicación

- **SOAP: Características físicas de la conexión.**

Existe una conexión a través del protocolo SOAP entre la PC cliente y el servidor de aplicaciones de la plataforma.

Conclusiones del Capítulo

Con el desarrollo de este capítulo queda expuesta la arquitectura definida para el sistema así como los patrones de arquitectura y diseño a utilizar. Mediante los diagramas de clases y de interacción quedan expuestas las clases asociadas a cada caso de uso y la manera en la que estas interactúan entre sí. Se muestra el despliegue del sistema en ejecución. Con la vista del más alto nivel conceptual ofrecida por la arquitectura y el menor nivel de abstracción y más detallado ofrecido por el diseño, se da guía y paso a la implementación.

CAPíTULO 4 IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

Introducción

En este capítulo se describe la implementación del sistema en términos de componentes y la manera en que estos componentes serán desplegados. Se ilustrarán los principales resultados obtenidos. Además, los casos de prueba para los principales casos de uso.

4.1 Modelo de Implementación

La implementación comienza a partir del resultado obtenido en el flujo de Análisis y Diseño y la mayor parte de este flujo tiene lugar en la fase de Construcción. El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos.

4.1.1 Diagrama de Componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Representa un fragmento de un sistema software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas. Los diagramas de componentes muestran los componentes del software y los artilugios por los que está compuesto, como los archivos de código fuente, las librerías o las tablas de una base de datos. Este tipo de diagrama muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables. Los componentes software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución.

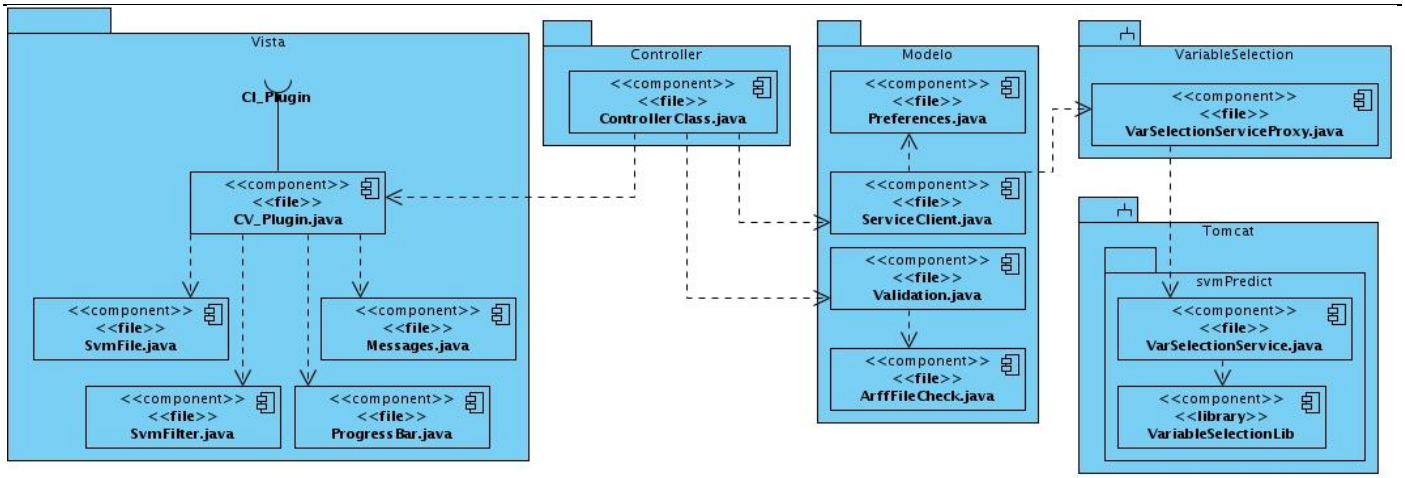

Figura 19. Diagrama de Componentes del CU Realizar Selección de Variables

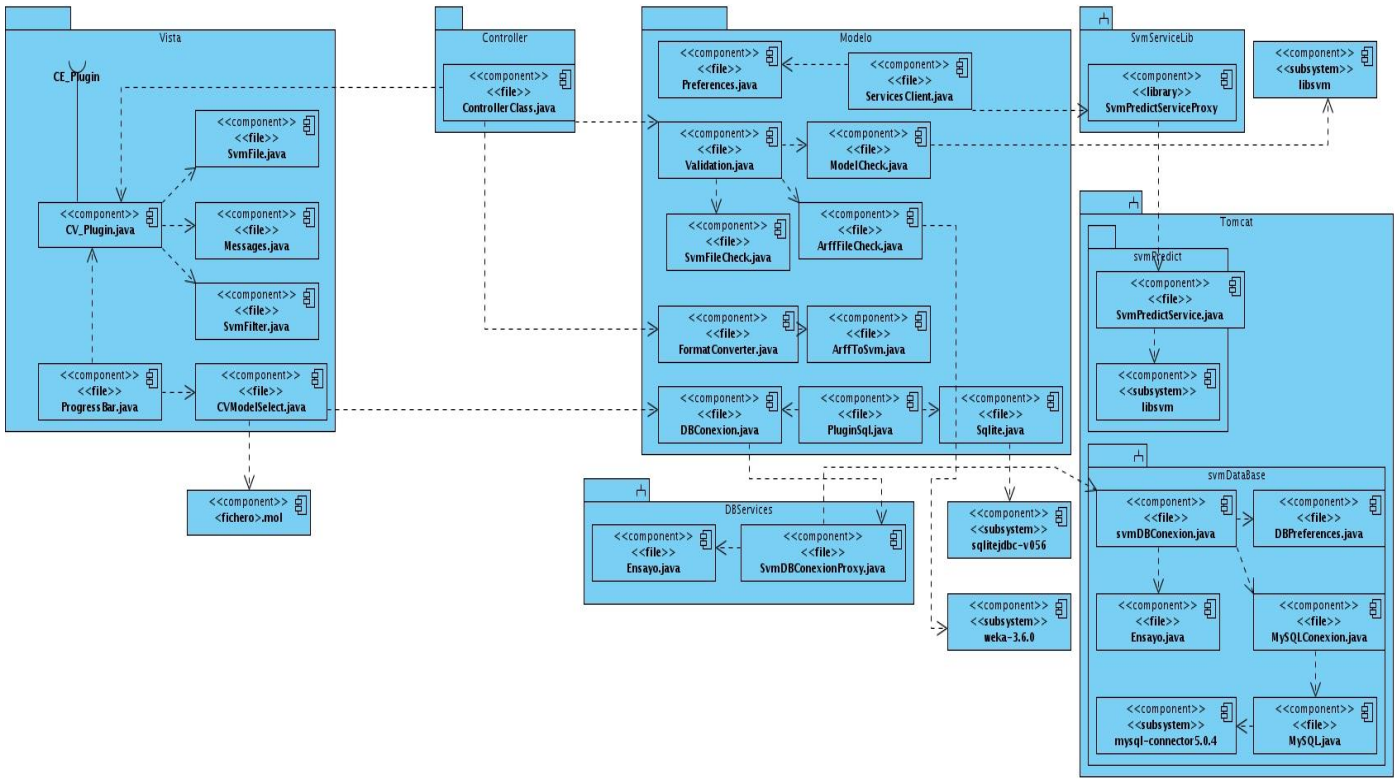

Figura 20. Diagrama de Componentes del CU Realizar Predicción

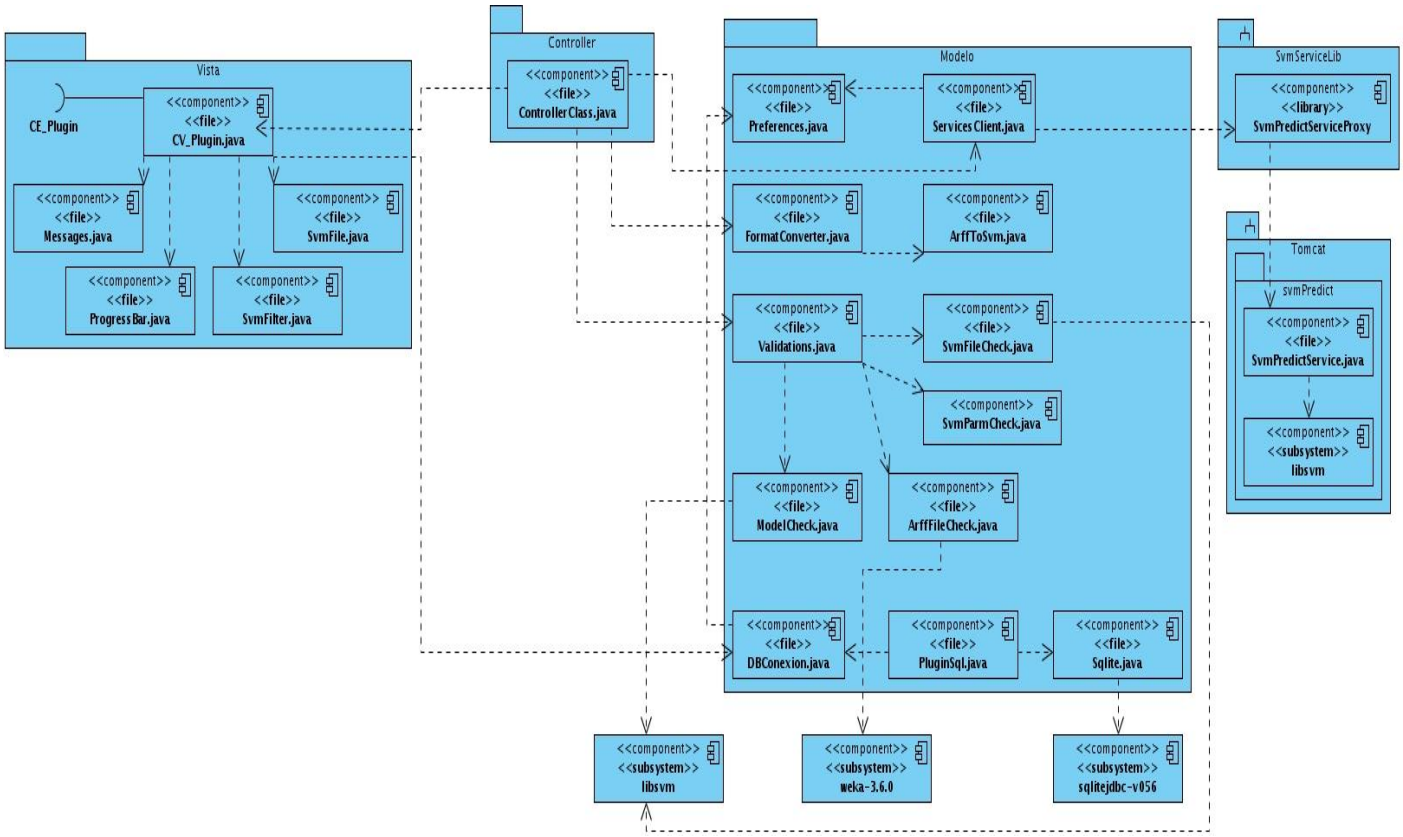

Figura 21. Diagrama de Componentes del CU Realizar Entrenamiento

4.1.2 Ejemplos de Código

El servicio de selección de variable recibe cómo parámetro un arreglo de tipo "String", indica qué método de búsqueda se usará, la medida de evaluación, así como los parámetros; además el contenido de un fichero de formato .arff que se reducirá. Luego de haber establecido los parámetros se llama al método "saveSolutions" de la librería de selección de variables. Finalmente se devuelve el fichero reducido.

```
/**
 * Web service operation
 */
@WebMethod(operationName = "varSelectionOperation")
public String varSelectionOperation(@WebParam(name = "data")
java.lang.String[] data) throws IOException, Exception {

    AttributeSelection attribute = null;

    ASEval eval = null;

    ASSearch search = null;

    logs("data 0 " + data[0]);
    logs("data 1 " + data[1]);
    logs("data 2 " + data[2]);
    logs("data 3 " + data[3]);
    logs("data 5 " + data[5]);

    switch (Integer.valueOf(data[0])) {
    case 0:
        writeFile("Hybrid", "parm", data[2]);
        ASSearch local = null;
        local = new GreedyStepwise();
        ((GreedyStepwise) local).setSearchBackwards(true);
        search = new HybridSearch();
        Vector<String> parameters = readHandG(new File(workDirectory,
                "Hybrid.parm"));
        ((GeneticSearch) search).setCrossoverProb(Double.valueOf(parameters
                .get(2)));
        ((GeneticSearch) search).setMutationProb(Double.valueOf(parameters
                .get(3)));
        ((GeneticSearch) search).setMaxGenerations(Integer
                .valueOf(parameters.get(1)));
           ((GeneticSearch) search).setPopulationSize(Integer
                   .valueOf(parameters.get(0)));
           ((HybridSearch) search).setLocalSearch(local);
           logs("configurÃ³ los datos hybridos");
           break;

       case 1:
           logs("SA alg...");
           writeFile("SA", "parm", data[2]);
           search = new SimulatedAnealing();
           Vector<String> parametersSA = readSA(new File(workDirectory, "SA.parm"));
           ((SimulatedAnealing) search).setInitTemperature(Double.valueOf(parametersSA.get(0)));
           ((SimulatedAnealing) search).setEndTemperature(Double.valueOf(parametersSA.get(2)));
           ((SimulatedAnealing) search).setIterations(Integer.valueOf(parametersSA.get(1)));
           ((SimulatedAnealing) search).setAlpha(Double.valueOf(parametersSA.get(3)));
           break;

       case 2:
           writeFile("SA", "parm", data[2]);
           Vector<String> parametersA = readHandG(new File(workDirectory,
           "SA.parm"));
           search = new GeneticSearch();
           ((GeneticSearch) search).setCrossoverProb(Double.valueOf(parametersA
                   .get(2)));
           ((GeneticSearch) search).setMutationProb(Double.valueOf(parametersA
                   .get(3)));
           ((GeneticSearch) search).setMaxGenerations(Integer
                   .valueOf(parametersA.get(1)));
           ((GeneticSearch) search).setPopulationSize(Integer
                   .valueOf(parametersA.get(0)));

           break;
       default:
           break;
```

```
Instances dataFile = new Instances(new BufferedReader(new FileReader(
        workDirectory.getAbsolutePath() + File.separatorChar + data[5]
                + ".arff")));
logs("echo inctancia");

logs("inicializando att");
attribute = new AttributeSelection(search, eval);
logs("terminado att");

logs("configurando parm att");
String[] param = data[3].trim().split(",");

if (param[0].equals("true")) {
    attribute.setMultipleOut(true);
} else {
    attribute.setMultipleOut(false);
}
logs("" + data[1]);
attribute.setCardinality(Integer.valueOf(param[1]));
logs("terminado con att ");
attribute.SelectAttributes(dataFile);
logs("terminaod de selecionar variables");
File reducedDirectory = new File(workDirectory, "Reduced");
if (!reducedDirectory.exists()) {
    reducedDirectory.mkdir();
}
logs("fichero de reducciÃ³n");
File reducedFile = new File(reducedDirectory, data[5] + ".arff");

RenderFile renderfile = new RenderArffFile(reducedFile
        .getAbsolutePath());
logs("salvande solucion");
attribute.saveSolutions(renderfile);
logs("solucion salvada");

String arffResult = readFile(reducedFile);
```

Figura 22. Ejemplo del método ¨varSelectionOperation¨

Al servicio predecir se le pasa por parámetro el contenido de un fichero de prueba, el fichero del modelo, así como la probabilidad de predicción. Se escribe en un fichero los datos de prueba y del modelo, además se crea un fichero donde se guardará la salida de la predicción. Se carga el objeto modelo del fichero para el modelo que se ha guardado, se verifica si el modelo soporta la probabilidad de predicción y luego se le pasan los parámetros a la librería de máquina de soporte vectorial, la cual retorna la predicción.

```
@WebMethod(operationName = "predict")
public java.lang.String[] predict(
        @WebParam(name = "inputString") String inputString,
        @WebParam(name = "modelString") String modelString,
        @WebParam(name = "predict_probability") int predict_probability)
        throws IOException {

    File input = new File(workDirectory, "input");
    File output = new File(workDirectory, "output");
    File model = new File(workDirectory, "model");

    writeFile(input, inputString);
    writeFile(output, "");
    writeFile(model, modelString);

    svm_model modelObj = svm.svm_load_model(model.getAbsolutePath());

    if (predict_probability == 1) {
        if (svm.svm_check_probability_model(modelObj) == 0) {
            System.err
                    .print("Model does not support probabiliy estimates\n");
            System.exit(1);
        }
    } else {
        if (svm.svm_check_probability_model(modelObj) != 0) {
            System.out
                    .print("Model supports probability estimates, but disabled in prediction.\n");
        }
    }

    String message = predict(input, output, modelObj, predict_probability);

    String[] result = { readFile(output), message };

    return result;
}
```

Figura 23. Ejemplo del método ¨predict¨

El servicio para el entrenamiento recibe los parámetros de la Máquina de soporte vectorial introducidas por el usuario, el fichero de prueba y el fichero donde se guardará el modelo del entrenamiento. Se lee el problema del fichero de prueba, luego se chequea los parámetros de la máquina de soporte teniendo cuenta el problema, si no hay errores entonces se le pasa los parámetros para el entrenamiento a la librería de Máquina de Soporte Vectorial, la cual nos devuelve el modelo del entrenamiento.

```
private void train(svm_parameter param, File inputFile, File modelOut) throws IOException {

    readProblem(inputFile);

    error_msg = svm.svm_check_parameter(prob, this.param);

    if (error_msg != null) {
        System.err.print("Error: " + error_msg + "\n");
        System.exit(1);
    }else {

        model = svm.svm_train(prob, param);
        model_file_name = modelOut.getAbsolutePath();
        svm.svm_save_model(model_file_name, model);
    }
}
```

Figura 24. Ejemplo del método ¨train¨

4.1.3 Pantallas principales de la aplicación

Las pantallas de la aplicación representan imágenes del sistema en pleno funcionamiento. Las aplicaciones desarrolladas constituyen los resultados principales del trabajo. A continuación se muestran estas aplicaciones.

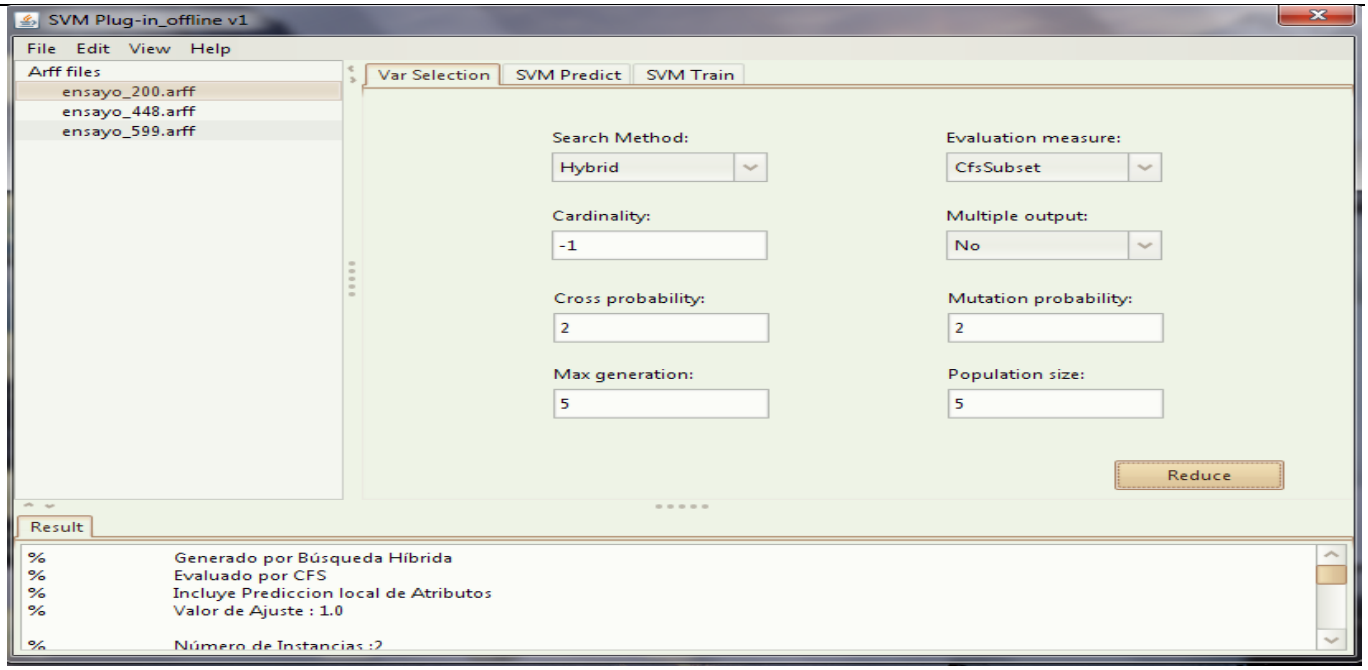

Figura 24. Interfaz para efectuar la Selección de Variables

En esta interfaz se realiza la selección de variables, el usuario carga los ficheros con formato arff, luego selecciona del árbol de ficheros al que le desea reducir las variables, escoge los métodos de búsqueda y las medidas de evaluación así como los restantes parámetros. Al presionar el botón reducir el sistema reduce las variables del fichero y le muestra al usuario en el área de resultados el resultado de la reducción.

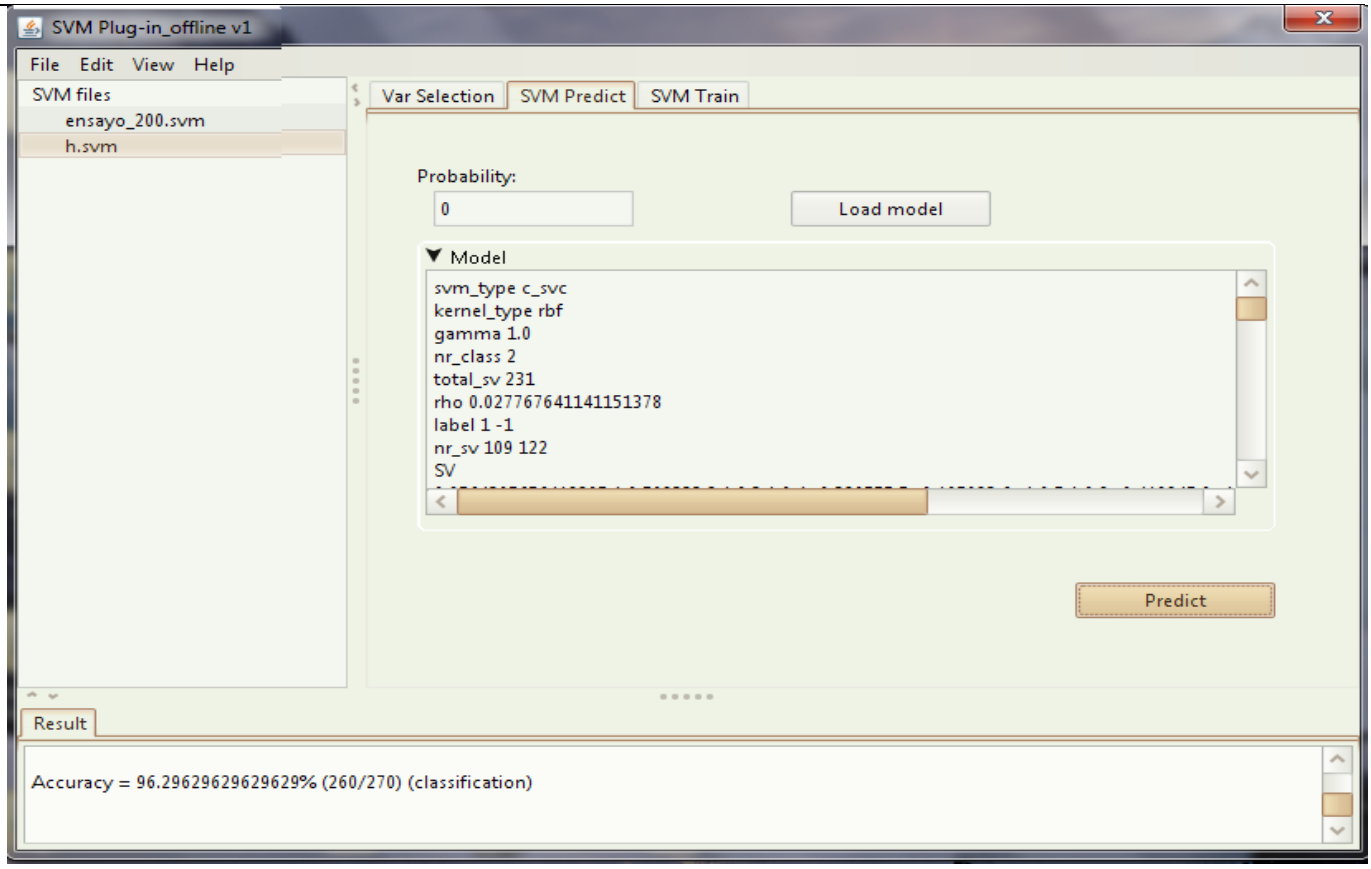

Figura 25. Interfaz de Predicción

Al usuario seleccionar la pestaña de predicción, se actualiza la interfaz permitiéndole al usuario introducir la probabilidad y cargar el modelo por el que se realizará la predicción; cuando este es cargado se le muestra al usuario si no es de la base de datos de alasGRATO. Después que el usuario selecciona el fichero de prueba presiona el botón predecir y el sistema le muestra la predicción en el área de trabajo. Si el modelo cargado fue el resultado de un entrenamiento por clasificación se le muestra el tanto por ciento, si fue por regresión se muestra el error cuadrático medio y el coeficiente de correlación cuadrático.

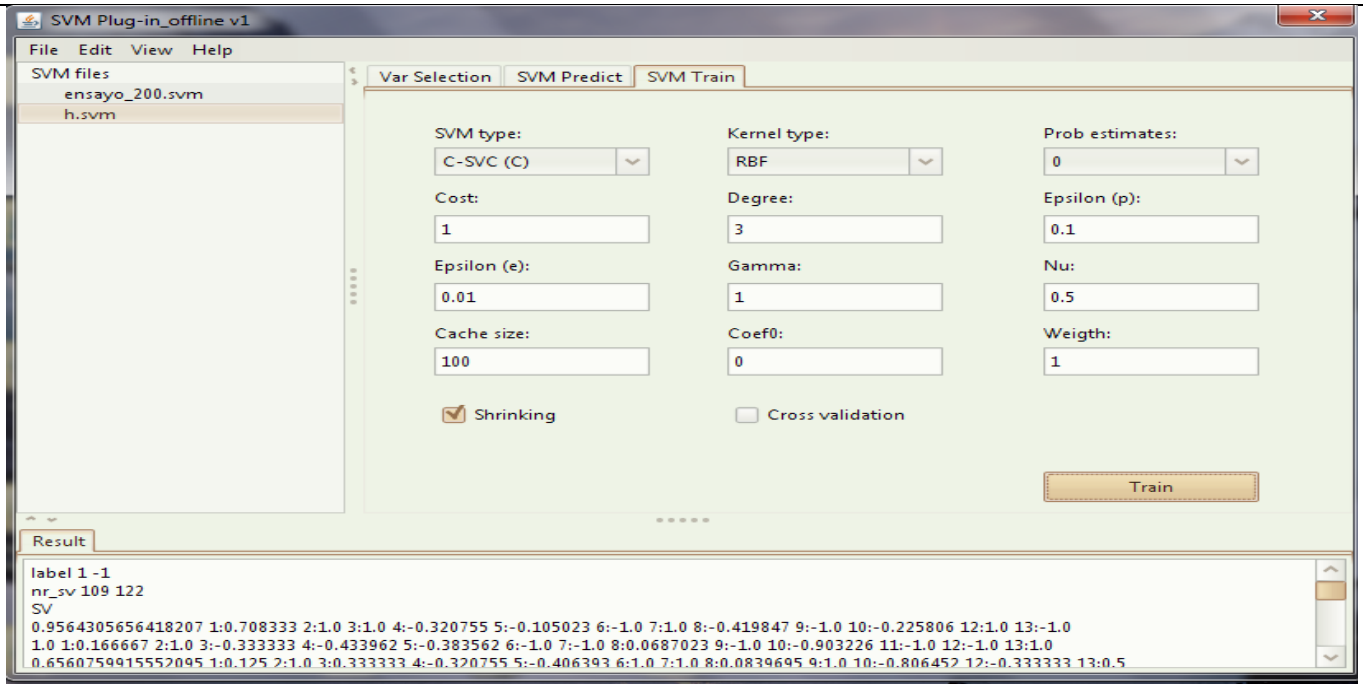

Figura 26. Interfaz # 1 de Entrenamiento

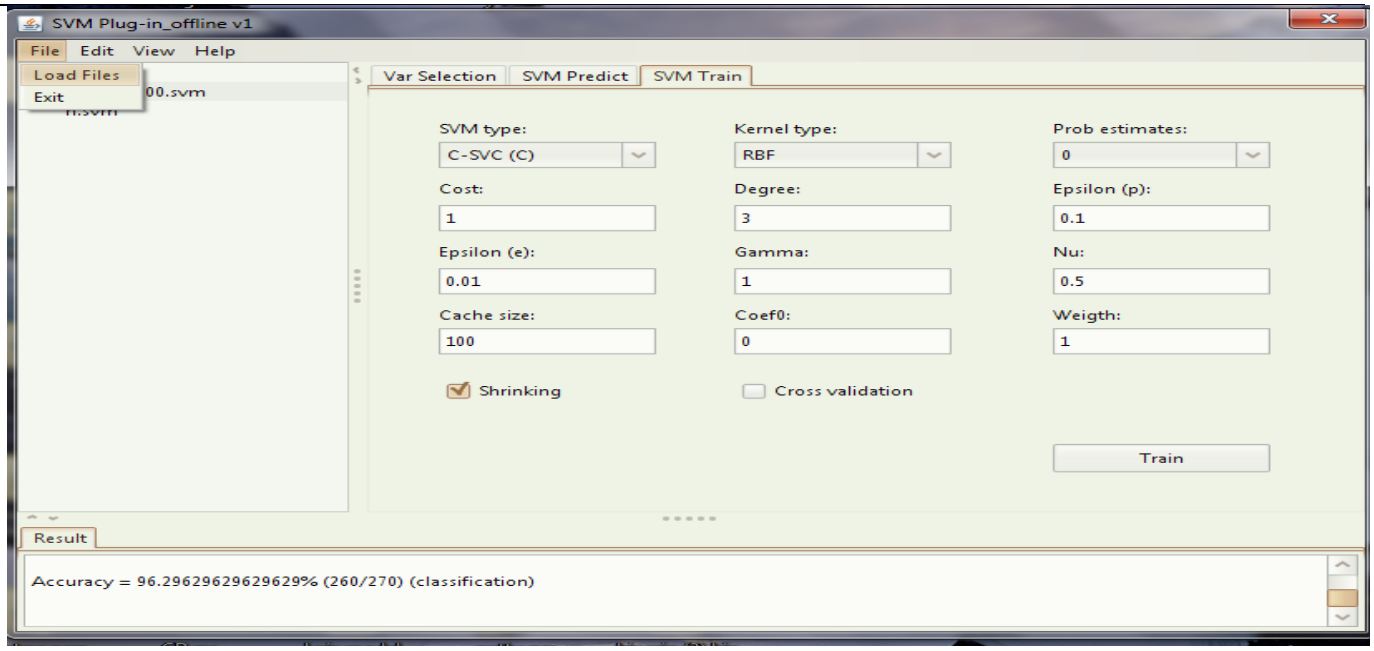

Figura 27. Interfaz # 2 de Entrenamiento

En la interfaz para el entrenamiento el usuario selecciona el fichero del árbol que desea entrenar, luego escoge el tipo de máquina de soporte vectorial y el tipo de kernel con el cual desea realizar en entrenamiento, además introduce los restantes parámetros. Al presionar el botón para entrenar el sistema entrena el fichero seleccionado con los parámetros escogidos y le muestra al usuario el modelo generado del entrenamiento.

4.2 Modelo de Pruebas

El flujo de trabajo de pruebas le presta servicios a los demás flujos. Su principal objetivo es evaluar o valorar la calidad del producto a través de la búsqueda y documentación de errores, validando el cumplimiento de requerimientos, el desempeño y dando una indicación de calidad. Las pruebas tienen éxito si descubre un error no detectado hasta entonces.

Pruebas de caja Negra

Las pruebas de caja negra son las que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Para validar las principales funcionalidades del sistema se utilizaron las pruebas de Caja Negra.

4.2.1 Casos de Pruebas

Los escenarios principales de los casos de uso críticos fueron probados para detectar las no conformidades. Se efectuaron las pruebas a 7 casos de pruebas, en la documentación del proyecto se representan estos casos de prueba. **(Ver planilla Diseño de Casos de Prueba)**

Conclusiones del Capítulo

Como resultado de este capítulo se obtuvo el diagrama de componentes por cada caso de uso, se hizo una representación de fragmentos de código con una breve explicación de los algoritmos más importantes que se implementaron y se mostraron las pruebas de caja negra realizadas con algunos de los resultados obtenidos.

CONCLUSIONES GENERALES

En el transcurso de octubre del 2009 y junio del 2010 fue desarrollado el presente trabajo, etapa en la cual se pudo arribar a un grupo de conclusiones consideradas como relevantes por los autores. En los primeros meses se llevó a cabo una importante revisión de la bibliografía disponible en pos de definir con claridad la estrategia a seguir en el desarrollo de la solución. Fundamentalmente obtenida desde Internet, debido a lo novedoso del tema tratado, la bibliografía consultada brindó una panorámica clara de las soluciones existentes hasta el momento y de los aspectos que de estas soluciones se debían conservar o superar. Finalmente se lograron cumplir los objetivos que se plantearon al inicio, expuestos de forma desglosada.

- Se analizó y diseñó una aplicación para la predicción de actividad biológica de compuestos orgánicos empleando Máquinas de Soporte Vectorial.

- Se implementaron dos plug-in, uno que realiza las tareas de forma local y otro plug-in que efectúa las tareas a través de servicios web para su incorporación a la primera versión del front-end de la plataforma.

- Se implementaron dos plug-in, uno que realiza las tareas de forma local y otro plug-in que efectúa las tareas a través de servicios web para su incorporación al front-end de la plataforma en su versión 2.0.

- Se realizó la evaluación de las principales funcionalidades del sistema haciendo uso de las pruebas de Caja Negra.

RECOMENDACIONES

- Desarrollar una segunda versión de la aplicación que incluya la programación distribuida usando la plataforma T-Arenal.
- Investigar el programa Weka para incorporarle los métodos de búsqueda y las medidas de evaluación que posee.
- Migrar los servicios a Axis2 para lograr que sean asincrónicos.

REFERENCIAS BIBLIOGRÁFICAS

1. Balmaseda, J. C. L. (2007) (activo Junio 2009). El Cáncer de los Cubanos. [En línea]. Disponible en: http://medicinacubana.blogspot.com/2007/04/el-cncer-de-los-cubanos.html

2. Steve, R. G. (1998) (activo Enero 2010): Support Vector Machines for Classification and Regression. [En línea]. Disponible en: http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM.pdf

3. Vapnik, V. N. (1998): Statistical Learning Theory. Springer, New York.

4. Steve, R. G. (1998) (activo Enero 2010): Support Vector Machines for Classification and Regression. [En línea]. Disponible en: http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM.pdf

5. Shawe-Taylor, J. y Cristianini, N. (2000) (activo Enero 2010). Support Vector Machines and other kernel-based learning methods. [En línea]. Disponible en: http://www.support-vector.net/

6. Pai-Hsuen, C., Chih-Jen, L. y Bernhard, S**.** (2009) (activo Enero 2010). A Tutorial on Nu-Support Vector Machines.

7. HERNÁNDEZ DÍAZ, Yaikiel. Desarrollo de modelos de clasificación de actividad biológica empleando máquinas de soporte vectorial*.* Febrero, 2010.

8. González, L. (2003). Modelos de Clasificación basados en Máquinas de Vectores Soporte. Editorial Asociación Española de Economía Aplicada, España, ISBN 84-607-7655-7.

9. Carreras, X., Márquez L. y Romero, E. (2004): Máquinas de Vectores Soporte. En: J. Hernández, M. Ramírez, y C. Ferri (eds), Introducción a la Minería de Datos, Editorial Pearson, España, 353-382 pp.

10. R, B.; M, T., Drug design by machine learning: support vector machines for pharmaceutical data analysis; [En línea]. [Consultado el: 15 de marzo de 2010]. Disponible en: http://citeseer.ist.psu.edu/528480.html.

11. VV, Z.; KV, B., Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions; [En línea]. [Consultado el: 15 de marzo de 2010]. Disponible en: http://pubs.acs.org/cgibin/abstract.cgi/jcisd8/2003/43/i06/abs/ci0340916.html

12. HX, L.; CX, X., Quantitative prediction of log of peptides in high-performance liquid chromatography based on molecular descriptors by using the heuristic method and support vector machine; [En línea]. [Consultado el: 18 de marzo de 2010]. Disponible en:http://pubs.acs.org/cgibin/abstract.cgi/jcisd8/2004/44/i06/abs/ci049891a.html

13. CX, X.; RS, Z., Study of the quantitative structure-mobility relationship of carboxylic acids in capillary electrophoresis based on support vector machines; [En línea]. [Consultado el: 10 de febrero de 2008]. Disponible en:http://pubs.acs.org/cgiin/abstract.cgi/jcisd8/2004/44/i03/abs/ci034280o.html

14. XUE, C. X.; ZHANG, R. S., An Accurate QSPR Study of O-H Bond Dissociation Energy in Substituted Phenols Based on Support Vector Machines; [En línea]. [Consultado el: 22 de enero de 2008]. Disponible en: http://pubs.acs.org/cgibin/abstract.cgi/jcisd8/2004/44/i02/abs/ci034248u.html

15. S, W.; C, X., Study on the quantitative relationship between the structures and electrophoretic mobilities of flavonoids in micellar electrokinetic capillary chromatography; [En línea]. [Consultado

el: 22 de diciembre del 2007]. Disponible en:http://www.cababstractsplus.org/google/abstract.asp?AcNo=20043056516

16. CY, Z.; RS, Z., QSAR study of natural, synthetic and environmental endocrine disrupting compounds for binding to the androgen receptor; [En línea]. [Consultado el: 10 de febrero de 2008]. Disponible en: http://www.informaworld.com/smpp/content~content=a727214684~db=all

17. S, Y.; M, X., Quantitative structure-property relationship studies on electrochemical degradation of substituted phenols using a support vector machine; [En línea]. [Consultado el: 23 de abril de 2008]. Disponible en:http://www.informaworld.com/smpp/content~content=a757824679~db=all~jumptype=rss

18. W, M.; F, L., Quantitative structure-property relationships for pesticides in biopartitioning micellar chromatograph; [En línea]. [Consultado el: 23 de enero de 2008]. Disponible en: http://www.aapspharmaceutica.com/search/view.asp?ID=74403

19. F, L.; W, M., Quantitative structure-activity relationship models for prediction of sensory irritants (logRD50) of volatile organic chemicals; [En línea]. [Consultado el: 23 de enero de 2008]. Disponible en: http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16307788&dopt=Abstract

20. IV LATIN AMERICAN CONGRESS ON BIOMEDICAL ENGINEERING; [En línea]. [Consultado el: 27 de abril de 2008]. Disponible en: http://www.claib2007.eventos.usb.ve/dmdocuments/Resumenes%20CLAIB%202007.pdf

21. Revista Facultad de Ingeniería Universidad de Antioquia; [En línea]. [Consultado el: 21 de marzo de 2010]. Disponible en: http://www.scielo.org.co/scielo.php?pid=S0120-62302009000100005&script=sci_arttext

22. CENATAV; [En línea]. [Consultado el: 21 de marzo de 2010]. Disponible en: http://www.cenatav.co.cu/es/investigadores/isneri_e.html

23. Página Oficial del Weka; [En línea]. [Consultado el: 21 de marzo de 2010]. Disponible en: http://www.cs.waikato.ac.nz/ml/weka/

24. Página de la librería LibSVM; [En línea]. [Consultado el: 21 de marzo de 2010]. Disponible en: http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16307788&dopt=Abstract.

25. Mallows, C. L. (1973): Some comments on Cp Technometrics. 15: 661-676 pp.

26. Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. Proc. 2nd International Symposium on Information Theory, Budapest: Akademia Kiado, 267-281 pp.

27. Schwarz, G. (1978): Estimating the dimension of a model. The Annals of Statistics: 6, 461-464 pp.

28. George, E. I., y Foster, D. P. (2000): Calibration and empirical bayes variable selection. Biometrika: 87 pp.

29. Garey, M. R. y Johnson, D. S. (1979) (activo Enero 2010). Computers and Intractability: a Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco, California. [En línea]. Disponible en: http://elib.tu-darmstadt.de/tocs/125242654.pdf

30. Servicios Web. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

31. Villaverde, Julio. *Un nuevo Front-End para la plataforma alasGRATO*. Tesis de Grado, Universidad de las Ciencias Informáticas, Cuba, 2009. [En línea]. [Consultado el: 10 de marzo del 2001]. Disponible en: http://biblioteca.uci.cu/cgi-

bin/biuci.exe?rec_id=007993&database=BIUCI&search_type=link&lang=spa&format_name=EFALL&page_header=EPHAV1.

32. Buschmann F, et al. Pattern-Oriented Software Architecture. John Wiley & Sons; 1996.

33. NARANJO, M. Fundamentos de Definición de Arquitectura de Software Bogotá, 2005. 2009.

34. *Patrones de Diseño en Aplicaciones Web con J2EE*. 2009. PDF. [En línea]. [Consultado el: 23 marzo 2010]. Disponible en: http://www.articuloweb.com/pdf.php?art_id=192

ANEXOS

Anexo 1: Búsqueda bibliográfica en google y EBSCO.

Palabra Clave	No. Registros Google	No. Registros EBSCO
Structure Activity Relationship	60 000 000	9300
Variable reduction	912 000	9164
Genetic Algorithms	94 600	8000
Hybrid Greddy	5 190	7324

GLOSARIO DE TÉRMINOS

A

Actividad biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos (Molécula o Fragmento).

B

Bioinformática: Es la aplicación de los ordenadores y los métodos informáticos en el análisis de datos experimentales y simulación de los sistemas biológicos.

C

Características: Atributos que posee cada una de las instancias.

Compuestos Orgánicos: Compuestos cuya composición fundamental es sobre la base del elemento químico carbono.

CASE: Computer Aided Software Engineering (Herramientas de ingeniería de software asistida por computadora).

D

Descriptor: Número que caracteriza estructuralmente la molécula.

E

Entrenamiento: Acción que se realiza para el aprendizaje de ciertas muestras.

Especialista: Cliente que utilizará el sistema.

F

Fragmento: Una pequeña parte de la molécula a la cual se llega a través de un método que es el encargado de fragmentarla.

M

Máquinas de Soporte Vectorial: Técnica de inteligencia artificial para la clasificación o regresión.

Metodología: Define quién hace qué, cómo y cuándo.

Modelo: Representación abstracta de la realidad.

Molécula: La partícula más pequeña de una sustancia, que mantiene las propiedades químicas específicas de esa sustancia, formada por un conjunto de átomos ligados por enlaces covalentes.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de software, que puedan funcionar en diversas plataformas.

P

Plataforma: Es el principio, ya sea de hardware o software, sobre el cual un programa puede ejecutarse.

Plug-in: Es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

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

S

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

V

Validación Cruzada: Método de estimación aleatorio que indica el por ciento de exactitud de los datos que serán correctamente clasificadas en una muestra.