Universidad de las Ciencias Informáticas

Facultad de Ciencias y Tecnologías Computacionales

"ProFeatPred: Sistema experto basado en modelos de rasgos de contacto bilineales para la predicción de propiedades biológicas en proteínas"

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

Autores:

Liliana Victoria Fonseca Aristigüi

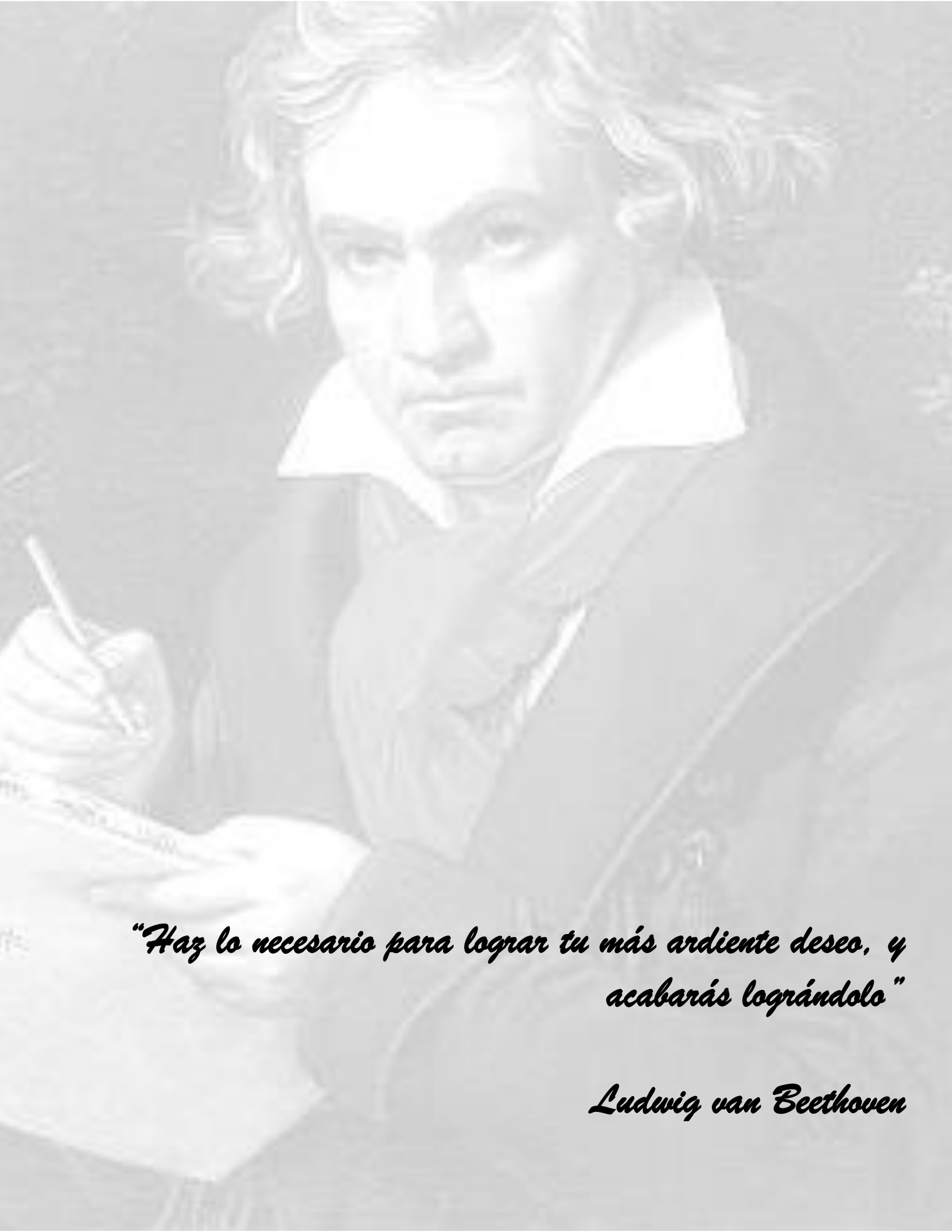
Lisdairy Sanabria Balber

Tutores:

M.Sc. Ernesto Contreras Torres

La Habana, Julio de 2017

"Año 59 de la Revolución"



"Haz lo necesario para lograr tu más ardiente deseo, y acabarás lográndolo"

Ludwig van Beethoven

Declaración de autoría

Declaramos ser autores de la presente tesis que tiene por título: ProFeatPred: Sistema experto basado en modelos de rasgos de contacto bilineales para la predicción de propiedades biológicas en proteínas 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 _____.

Liliana Victoria Fonseca Aristigüi

_____

Firma del Autor

Lisdairy Sanabria Balber

_____

Firma del Autor

M.Sc. Ernesto Contreras Torres

_____

Firma del Tutor

Datos de contacto

Autores:

Liliana Victoria Fonseca Aristigüi

Universidad de Ciencias Informáticas

Email: lilianav@uci.cu

Lisdairy Sanabria Balber

Universidad de Ciencias Informáticas

Email: lbalber@estudiantes.uci.cu

Tutor:

M.Sc., Ing., Ernesto Contreras Torres

Especialidad de graduación: Bioinformática y Biología Computacional (UCLV), Ciencias Informáticas (UCI).

Categoría docente: Instructor

Categoría científica: Máster

Años de experiencia en el tema: 4

Años de graduado: 4

Email: econtreras@uci.cu

A Mami y Tavio, por educarme bajo los principios de la responsabilidad, el respeto, la constancia y el sacrificio por aquello que deseo alcanzar dándome la fuerza para continuar con mis estudios y por las noches que estuvieron pidiendo a Dios para que me cuidara.

A toda mi familia, en especial a tio Vachy, tia Julio, tia Xiomara, Neysi, Dailén y mis hermanos, que han seguido cada paso de mi vida estudiantil, alegrándose, sufriendo, alentándome y dándome consejos para mejorar.

A todos mis amigos, a Yeni, por ser la mejor amiga del mundo, por siempre estar ahí, por regañarme, cuidarme, siempre estar conmigo en los momentos más difíciles de mi vida y ser como una hermana mayor para mí.

A todo mi turno de trabajo por apoyarme, siempre los llevaré en el corazón a donde quiera que vaya y me acompañarán los recuerdos de los momentos vividos juntos.

A cada uno de mis compañeros de aula, que fueron mis compañeros de viaje en esta desgastante carrera.

A mi tutor por toda la ayuda, paciencia y esfuerzo brindado. Sin él este trabajo no hubiera sido posible.

Y mis profesores que he tenido a lo largo de estos seis años ya que soy el resultado esmerado del trabajo de cada uno de ellos.

Liliana

Es imposible plasmar todos los nombres de las personas que están ligadas indisolublemente a nuestras vidas, agradezco a todos los que me han apoyado, respaldado, animado y han reído conmigo en estos seis años.

Especialmente agradecer a mis padres, por ser guía e inspiración en todo momento, por contribuir en mi formación y por toda la confianza que depositaron en mí, sin ellos nada hubiese sido posible.

A mis hermanos por darme su apoyo y en especial a Lemay por ser mi guía en todo momento, por haberme defendido siempre y haberme corregido cuando fue necesario.

Agradecer a mi tutor por su guía constante, por dedicar parte de su tiempo a asesorarme, orientarme y aconsejarme.

A mis amigos gracias por estar siempre presentes en las buenas y en las malas, gracias por el apoyo que me brindaron, y el granito de arena que aportó cada uno, de una manera u otra,

A mi esposo que ha estado a mi lado ayudándome, apoyándome y alentándome durante el desarrollo de este trabajo. A mi niño por ser mi fuente de inspiración y darme la fuerza para seguir a pesar de que existían dificultades.

A aquellos profesores que en estos años de estudio me educaron y formaron como profesional,

A todos gracias…

Lisdairy

Dedicatoria

Este trabajo está dedicado a las personas más especiales en mi vida: Mami y Tavio, por darme su amor, su apoyo y por guiarme en el camino de la vida.

Liliana

Dedico este trabajo a mi mamá y mi papá por haber estado siempre presente en mi vida y a mi niño por ser mi razón de ser

Lisdairy

Resumen

En la actualidad para caracterizar y predecir las propiedades de las biomoléculas se requiere de una investigación de varios años y cuantiosos recursos financieros. Por lo tanto, es de suma importancia la identificación de las dianas biológicas en el proceso de desarrollo de un determinado fármaco.

En el presente trabajo se propone el uso de una herramienta informática que posibilita la disminución del tiempo de investigación y la de inversiones económicas. Primeramente se desarrolla un módulo para el cálculo de descriptores 3D-proteicos. Este módulo esta implementado en el software ToMoCoMD-CAMPS. El cual se basa en la aplicación de las formas algebraicas bilineales sobre la matriz de contacto. Esta matriz se utiliza para codificar información relativa a las interacciones no covalentes entre aminoácidos. Con este objetivo se aplican las matrices simple-estocástica y de probabilidad mutua para normalizar la matriz de contacto no estocástica. Además, se desarrolla un sistema experto basado en modelos para predecir la velocidad de plegamiento y la clase estructural de proteínas. La herramienta que se propone es un sistema experto que emplea las técnicas de Regresión Lineal Múltiple y Perceptrón Multicapa. Ambas técnicas permitieron obtener modelos robustos con buena capacidad predictiva. Esta herramienta informática posibilita el conocimiento del resultado de forma anticipatoria durante la investigación y permite la aplicación de los sistemas desarrollados como herramientas complementarias a los enfoques existentes en la modelación de propiedades biológicas y/o funciones de interés en proteínas.

Palabras claves: contacto, descriptor 3D-proteico, formas bilineal, Perceptrón Multicapa, Regresión Lineal Múltiple, sistema experto, ToMoCoMD-CAMPS.

Abstract

Nowdays to characterize and predict the properties of biomolecules it is requiered some years of investigation work and substantial financial resources. Because of this, it is very ,important the identification of biological targets in the development process of a given drug.

In the present work is proposed the use of a informatics tool to decrease the time and econonical investments. First, it is developed a module to calculate 3D-protein descriptors. This tool es implemented in ToMoCoMD-CAMPS software. It is based on the application of bilinear algebraic forms on the contact matrix. With this objective it is used to encode information regarding non-covalent interactions between amino acids. The simple-stochastic and mutual-probability matrices were managed to normalize the non-stochastic contact matrix. In addition, a model-based expert system is developed to predict folding rate and structural class of proteins. The proposal tool is an expert system that uses Multiple Linear Regression and Multi-layer Perceptron techniques. Both tecniques may posible to obtain robust models with good predictive capacity were obtained. As a result, the proposed application alows to know ahead of time the result of biological targets and the use of the systems developed as complementary tools to the existing approaches in the modeling of biological properties and/or functions of interest in proteins.

Keywords: contact, 3D-protein descriptor, bilinear forms, Multi-layer Perceptron, Multiple Linear Regression, expert system, ToMoCoMD-CAMPS.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Introducción

El desarrollo de una terapia para determinada patología es un proceso comúnmente constituido por tres pasos. El primer paso es la identificación de la diana biológica o terapéutica, es decir, la identificación de una molécula biológica, principalmente proteínas, involucrada en algún mecanismo implicado en algún proceso patológico. Un estudio relativamente reciente desarrollado por el *Boston Consulting Group* y que implicó a 50 compañías e instituciones académicas, reveló que el proceso de desarrollo de un nuevo medicamento hasta su uso autorizado en terapéutica requiere, en promedio, la inversión de 880 millones de dólares (USD) y 15 años de investigación (Borroto et al. 2013). Lo anterior evidencia la extrema complejidad asociada a la tarea de desarrollar "un nuevo medicamento", pero también muy valorada por la sensibilidad que genera el impacto negativo de las enfermedades en la sociedad moderna.

Una alternativa para la identificación de este tipo de moléculas son los denominados estudios QSAR (Quantitative Structure-Activity Relationships) pues permiten estimar, con aceptable grado de precisión, la actividad/propiedad de nuevos compuestos, por lo que pueden aplicarse como estrategia de tamizaje virtual como alternativa a los costosos procesos de síntesis y bioensayos (Tropsha 2010).

Notables esfuerzos se han dedicado a la codificación estructural de pequeñas moléculas orgánicas como se evidencia en (Todeschini and Consonni 2000; Todeschini and Consonni 2009). Sin embargo, la búsqueda de nuevos descriptores y nuevas representaciones de proteínas constituye un área de creciente importancia en ciencia de proteínas (González-Díaz et al. 2007; González-Díaz et al. 2008; Randic et al. 2009) debido no solo al creciente volumen de datos biológicos, sino también a su probada utilidad en la predicción de propiedades o funciones biológicas cuando se combinan con técnicas estadísticas y de aprendizaje automático (Chou 2011; Chou 2015). De especial importancia resultan los métodos que abordan la caracterización de la conformación espacial (3D) de proteínas, pues como es bien conocido, la misma está relacionada con la función que realizan estas estructuras químicas en los organismos vivos (Boyle 2005). Ejemplos hay varios: descriptores de ProtDCal (Ruiz-Blanco et al. 2015b), índice I3 (Estrada 2002), los descriptores markovianos (Humberto González-Díaz1 and Uriarte1 2008; Humberto González-Díaz* 2007), y las denominadas redes de contacto (L. Di Paola 2012). Trabajos recientes ponderan a las redes de contacto como un enfoque apropiado para la descripción de la estructura tridimensional de proteínas (Brinda and Vishveshwara 2005; da Silveira et al. 2009; Di Paola et al. 2012; Paola et al. 2016; Yan et al. 2014; Zhou et al. 2014) . En general, estos métodos representan la geometría proteica como una matriz binaria de una red (grafo) de las interacciones (contactos) que se establecen entre los aminoácidos que conforman las proteínas.

Por otra parte, en (Contreras-Torres et al.), se proponen descriptores 3D-proteicos basados en la aplicación de las formas algebraicas bilineales sobre la matriz de distancia Minkowski, los cuales se utilizaron de forma

satisfactoria para el reconocimiento de las clases estructurales de proteínas y encuentran disponibles en el software de libre acceso ToMoCoMD-CAMPS (Torres 2016). Resulta importante señalar que la matriz de distancia Minkowski es continua, es decir, que sus respectivas entradas son números reales, los cuales representan la distancia entre pares de aminoácidos. Por lo que resulta interesante la definición nuevos descriptores de contacto en el marco de las formas bilineales y su aplicación en la predicción de propiedades biológicas en proteínas.

Finalmente, en el lineamiento 131 de la Política Económica y Social del Partido y la Revolución se plantea lo siguiente: "Sostener y desarrollar los resultados alcanzados en el campo de la biotecnología, la producción médico-farmacéutica,…, las ciencias básicas,…, los servicios científicos y tecnológicos de alto valor agregado" (Acebedo 2012).

Por lo anteriormente expresado, se plantea el siguiente **problema de investigación**: ¿Cómo contribuir a la identificación de propiedades biológicas en proteínas? Para darle solución al problema anterior se define como **objetivo general**: desarrollar un sistema experto basado en modelos de rasgos de contacto bilineales para la predicción de propiedades biológicas en proteínas. A partir del problema antes planteado se define como **objeto de estudio**: sistemas expertos basados en modelos, enmarcado en el siguiente **campo de acción**: sistema experto basado en modelos de rasgos contacto bilineales para la predicción de propiedades biológicas en proteínas. El objetivo general fue desglosado en los siguientes **objetivos específicos**:

- ✓ Establecer la selección de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema experto basado en modelos.
- ✓ Realizar el análisis y diseño del sistema experto basado en modelos.
- ✓ Implementar el sistema experto basado en modelos.
- ✓ Realizar pruebas al sistema experto basado en modelos.

Para cumplir los objetivos específicos se proponen las siguientes **tareas de investigación**:

- ✓ Revisión bibliográfica sobre los sistemas expertos usados en estudios bioinformáticos.
- ✓ Realización del análisis y diseño del módulo para el cálculo de los descriptores de contacto (MDC).
- ✓ Implementación del MDC.
- ✓ Evaluación del MDC.
- ✓ Desarrollo de modelos para predecir clase estructural.
- ✓ Desarrollo de modelos para predecir velocidad de plegamiento.
- ✓ Realización del análisis y diseño del Sistema Experto (SE).
- ✓ Implementación del SE.

- ✓ Evaluación del SE.

Luego de la realización de la revisión bibliográfica se formuló la siguiente **hipótesis de investigación**: Si se utiliza un sistema experto basado en modelos desarrollado con descriptores de contacto bilineales, entonces es posible predecir velocidad de plegamiento y clase estructural de proteínas.

En el desarrollo del presente trabajo se utilizan los siguientes **métodos de investigación**:
Métodos teóricos:

- ✓ **Analítico-Sintético**: Este método permite analizar la bibliografía existente sobre los sistemas expertos, identificando los elementos similares, extrayéndolos y sintetizándolos, contribuyendo de esta forma a la solución del problema planteado.
- ✓ **Histórico-Lógico**: A través de este método se pudo analizar el desarrollo de los sistemas expertos y cómo han evolucionado en el campo de la bioinformática.
- ✓ **Modelación**: Este método permite la elaboración de los diferentes diagramas que se utilizarán en la investigación y se aplica además para generar los artefactos del proceso de ingeniería de software, sirviendo de base para la implementación del sistema.

Métodos empíricos:

- ✓ **Análisis documental**: Permite revisar los documentos, artículos, libros y revistas especializadas en los temas referentes a la investigación.

Técnicas de recopilación de información

- ✓ Tormenta de ideas: Se emplea con el objetivo de identificar las características y capacidades con las que debe cumplir el sistema experto basado en modelos para la predicción de propiedades biológicas en proteínas, teniendo en cuenta los criterios del tutor y los autores.

Capítulo 1: Fundamento teórico sobre el desarrollo de sistemas expertos

En el capítulo se abordan los principales conceptos relacionados con la Inteligencia Artificial, los Sistemas Expertos, su arquitectura, sus principales ventajas y desventajas, así como la metodología a utilizar. Se muestran ejemplos de sistemas expertos en la bioinformática y se abordan los métodos estadísticos y de aprendizaje automáticos utilizados en los estudios realizados. Además, se abordan los descriptores de contacto y se presenta su definición matemática. Finalmente, detallan las herramientas, tecnologías y lenguaje de programación utilizadas para el desarrollo de ProFeatPred.

1.1. Inteligencia Artificial

El término Inteligencia Artificial (IA) se refiere a la capacidad de emular las funciones inteligentes del cerebro humano (Badaró et al. 2013). Esta es una de las disciplinas de la ciencia y la ingeniería de más reciente surgimiento. A través de ella muchas representaciones y métodos que al parecer la gente utiliza de manera inconsciente se han concretado. Se ha definido “la inteligencia artificial se considera una rama de la computación y relaciona un fenómeno natural con una analogía artificial a través de programas de computador. La inteligencia artificial puede ser tomada como ciencia si se enfoca hacia la elaboración de programas basados en comparaciones con la eficiencia del hombre, contribuyendo a un mayor entendimiento del conocimiento humano” (Alvarez 1991). Aunque han sido muchos los términos definidos hasta el momento sobre la IA todos ellos concuerdan en que es la disciplina que se ocupa de diseñar y construir sistemas inteligentes que simulen el comportamiento humano.

La pieza comercial y la que más aplicación se le ha dado en IA es a los sistemas expertos (Carlos Soto 2005), estos son considerados como un subconjunto de la IA (Rossini 2000).

1.2. Sistemas Expertos

Un Sistema Experto (SE) es un sistema computacional que adquiere conocimiento especializado en un campo específico para explotarlo mediante métodos de razonamiento que emulan el desempeño del experto humano en la solución de problemas (Peña Ayala 2006).

Otros autores han definido a los SE como “programas sofisticados de computación que tratan de imitar funciones de un experto en algún dominio del conocimiento, es decir, manipulan conocimientos de expertos para resolver eficiente y efectivamente, problemas de un área específica, tal como lo hacen los expertos humanos. Incorporan operativamente el conocimiento de una o varias personas experimentadas. Resuelven problemas de forma “inteligente” y son capaces de explicar y justificar sus respuestas. Son creados para actuar como asistentes “inteligentes” en los procesos de tomas de decisión” (Habana 1991).

1.2.1. Arquitectura de los sistemas expertos

Se encuentran variedad de bibliografías que detallan diversas arquitecturas para un SE, todas concuerdan en los principales elementos. En la Fig. 1 se muestra un módulo de componentes más general que se puede utilizar en cualquier ambiente de trabajo.

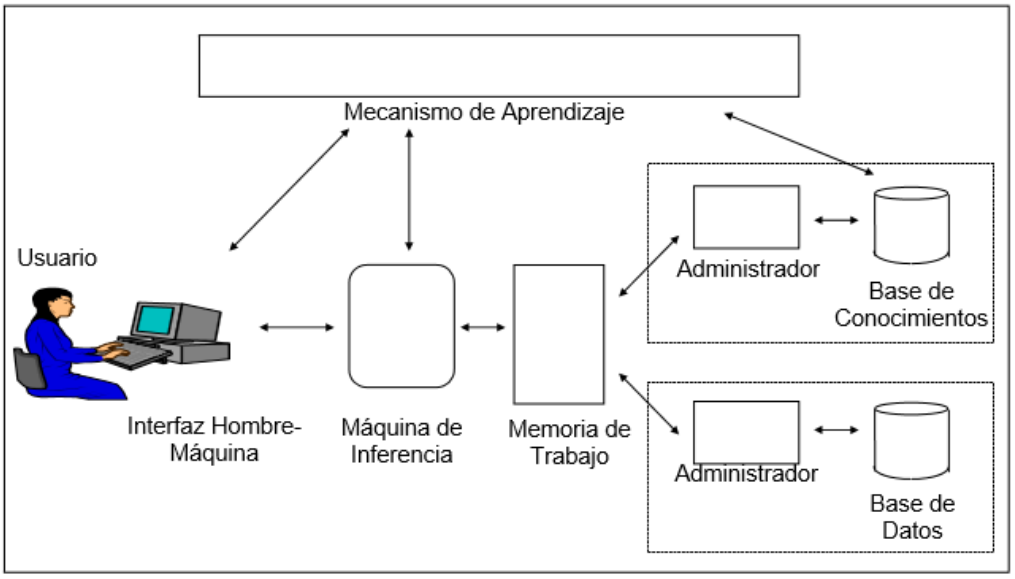


Fig. 1 Arquitectura general de los componentes de un Sistema Experto (Peña Ayala 2006)

Mecanismo de aprendizaje: Es el módulo responsable de adquirir nuevo conocimiento y actualizar el existente, descomponiendo a los subsistemas interfaz hombre-máquina, máquina de inferencia y memoria de trabajo.

Usuario: Utiliza el sistema en cualquier modalidad como investigador, cliente, profesor o alumno.

Interfaz Hombre-Máquina: Mediante ella el usuario plantea los problemas al SE, establece el protocolo de diálogo, además explica el comportamiento del sistema.

Máquina de inferencia: Conocido como motor de inferencia, determina cuáles son las reglas aplicables en cada momento y se encarga de ejecutarlas.

Memoria de trabajo: Es donde se almacena el código, conocimiento y resultado de las inferencias que se generan, se desarrolla la solución del problema.

Base de conocimiento: Constituye el conjunto de conocimiento especializado del sistema experto.

Base de datos: Es una memoria de trabajo que alberga los datos propios de los problemas que se desean tratar, representación y manipulación, se concibe como dato o información.

1.2.2. Ventajas y desventajas de los sistemas expertos

Los SE permiten ofrecer soluciones técnicas más completas, por esta razón los mismos constituyen un nivel especializado en la representación y explotación de aplicaciones basadas en conocimiento. A diferencia de un experto humano los sistemas expertos no envejecen, no se ven afectados por condiciones externas y tampoco sufren pérdidas con el paso del tiempo. Otras ventajas de los sistemas expertos es que tienen alto grado de efectividad en la resolución de problemas, genera múltiples soluciones por contemplar varias hipótesis, emplea generalmente interfaz de lenguaje natural e interactúa con el humano o con el medio que controla (Peña Ayala 2006).

Como todas las aplicaciones informáticas los SE tienen limitaciones, una de ellas es que para actualizarlos hay que reprogramarlos. Los mismos no poseen sentido común para resolver situaciones complejas ni pueden controlar situaciones ambiguas, para ellos no hay nada obvio, un experto humano puede mantener una conversación informal y es capaz de distinguir cuales son las cuestiones relevantes de un problema y separarlas de cuestiones secundarias, mientras que un SE no puede. Cualquier persona aprende con relativa facilidad de sus errores y de errores ajenos, para estos sistemas es muy complejo hacerlo, por lo que es restringido en su dominio. Otra de sus limitaciones es que pueden ser muy costosos en cuanto a tiempo y dinero, son pocos flexibles a cambios y de difícil acceso a información no estructurada (Badaró, Ibañez and Agüero 2013) .

1.2.3. Metodologías de los sistemas expertos

Para el desarrollo de un sistema experto, al igual que se utiliza en la construcción de cualquier otro software informático convencional se emplea una metodología de desarrollo para guiar la confección del mismo. En el caso de los sistemas expertos se está en presencia de un área de desarrollo relativamente novedosa, por lo que no se utiliza una metodología en específico, sino que cada autor propone la que para él es más conveniente según la forma en que desarrolla el producto. En la actualidad hay metodologías que han tenido más aceptación que otras por lo que se han difundido.

De forma general se abordan algunas de estas metodologías enfatizando en la que se utiliza para el desarrollo de esta aplicación, la metodología de John Durkin.

1. Metodología de Buchanan
2. Metodología de Grover
3. Metodología KADS
4. CRISP-DM
5. Metodología de John Durkin

Metodología de Buchanan

Esta metodología fue creada por Bruce G. Buchanan (Buchanan et al. 1983). La característica más notable es la constante relación que debe existir entre el Ingeniero de conocimiento y el Experto Humano, genera una amplia documentación, pues está enfocada en la Adquisición del Conocimiento, la misma está pensada para el desarrollo de grandes sistemas expertos en los cuales se consta de un equipo de desarrollo numeroso.

Consta de cinco etapas fundamentales las cuales se muestra en la Fig. 2.

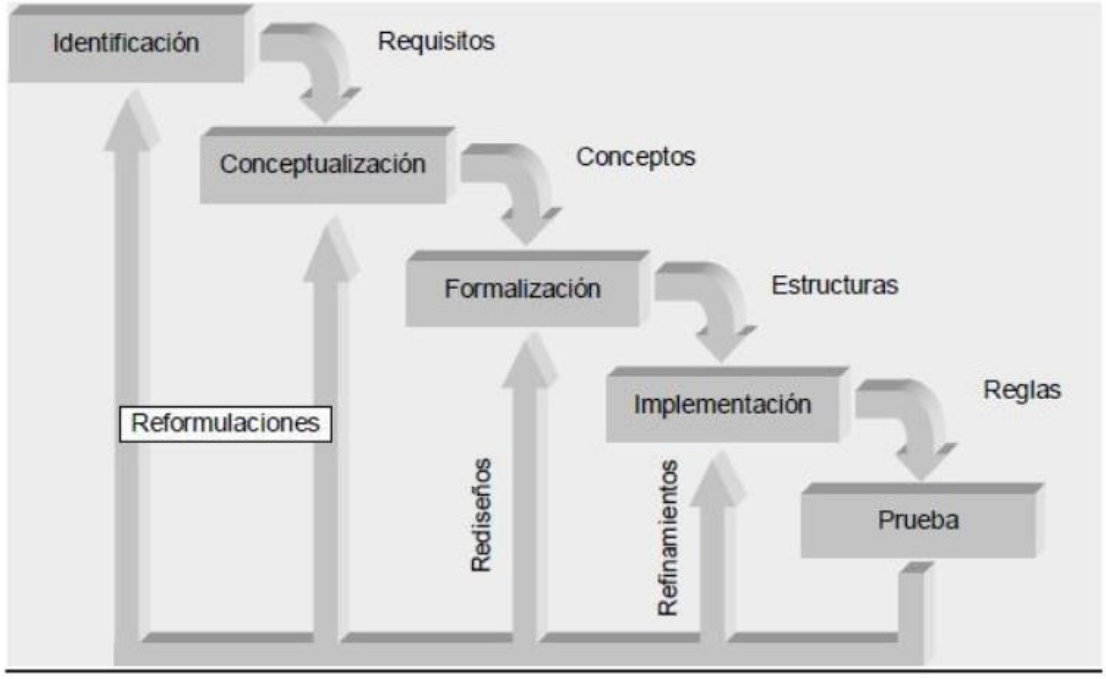


Fig. 2 Etapas de la metodología Buchanan

Metodología de Grover

Esta metodología fue creada en el 1983 se enfoca en el desarrollo del proceso de Adquisición del Conocimiento (Grover 1983). La característica más importante es el énfasis en la obtención de documentación, la cual reemplaza parcialmente al experto y además otorga referencia a los diseñadores y usuarios del sistema, presenta tres fases fundamentales, la definición del dominio, donde se realiza una descripción general del problema, se hace toda la revisión bibliográfica, se identifica al experto o los expertos y las distintas métricas para evaluar el rendimiento del sistema. La segunda, formulación del conocimiento general, tiene como objetivo examinar los escenarios a partir de criterios de evaluación y reclasificarlos, y una tercera donde se hace una revisión y ciclos de corrección. Esta metodología se basa en el típico ciclo de vida en cascada utilizado en los inicios de la ingeniería, de la que se puede deducir que el proceso de

desarrollo de un sistema experto se plantea como un proceso de revisión casi constante, está pensada para un equipo de desarrollo relativamente grande.

Metodología KADS

Esta metodología parte siendo un método para la Adquisición del Conocimiento la cual fue denominada KADS, posteriormente se amplió al desarrollo completo de Sistemas Expertos (desde el análisis y diseño del software hasta la gestión del proyecto), la cual se conoce actualmente con el nombre de CommonKADS. Fue una propuesta del Instituto de Tecnología de Massachusetts (Akkermans et al. 1999).

Esta metodología posee dos grandes características, la gestión de proyecto que involucra aspectos administrativos los cuales generalmente no son considerados al momento de desarrollar un sistema informático y el planteamiento de desarrollo de modelos que refleja las diferentes vistas del proyecto, estos modelos son: Diseño, Conocimientos, Comunicación, Organización, Tareas y Agentes. Está basada en el ciclo de vida en espiral donde al final de cada etapa se entrega la documentación apropiada antes de pasar a la siguiente. Aunque cubre todos los aspectos para ejecutar un proyecto de desarrollo de un SBC, presenta algunos problemas en cuanto a su aplicación, entre los que se encuentran: es una metodología muy amplia y compleja, no existe una fuente de información que contenga todo lo necesario para su aplicación, existen ejemplos parciales de su aplicación, pero no un ejemplo completo a utilizar como guía.

A continuación, se nombran las etapas de la metodología:

- ✓ Análisis
- ✓ Diseño
- ✓ Implementación
- ✓ Uso
- ✓ Mantenimiento y refinamiento del conocimiento

CRISP-DM

Según la encuesta publicada en octubre de 2014 por el sitio especializado KDNuggets, CRISP-DM (Cross Industry Standard Process for Data Mining) fue votada como la metodología más utilizada en el desarrollo de proyectos de descubrimiento de conocimiento, con un 43% de uso. Los orígenes de CRISP-DM, se remontan hacia el año 1999 cuando un importante consorcio de empresas europeas tales como NCR (Dinamarca), AG (Alemania), SPSS (Inglaterra), OHRA (Holanda), Teradata, SPSS, y Daimer-Chrysler, proponen a partir de diferentes versiones de KDD (Knowledge Discovery in Databases) el desarrollo de una guía de referencia de libre distribución denominada CRISP-DM (Chapman et al. 2000). Las fases del desarrollo de esta metodología son seis y se muestran en la siguiente Fig. 3.

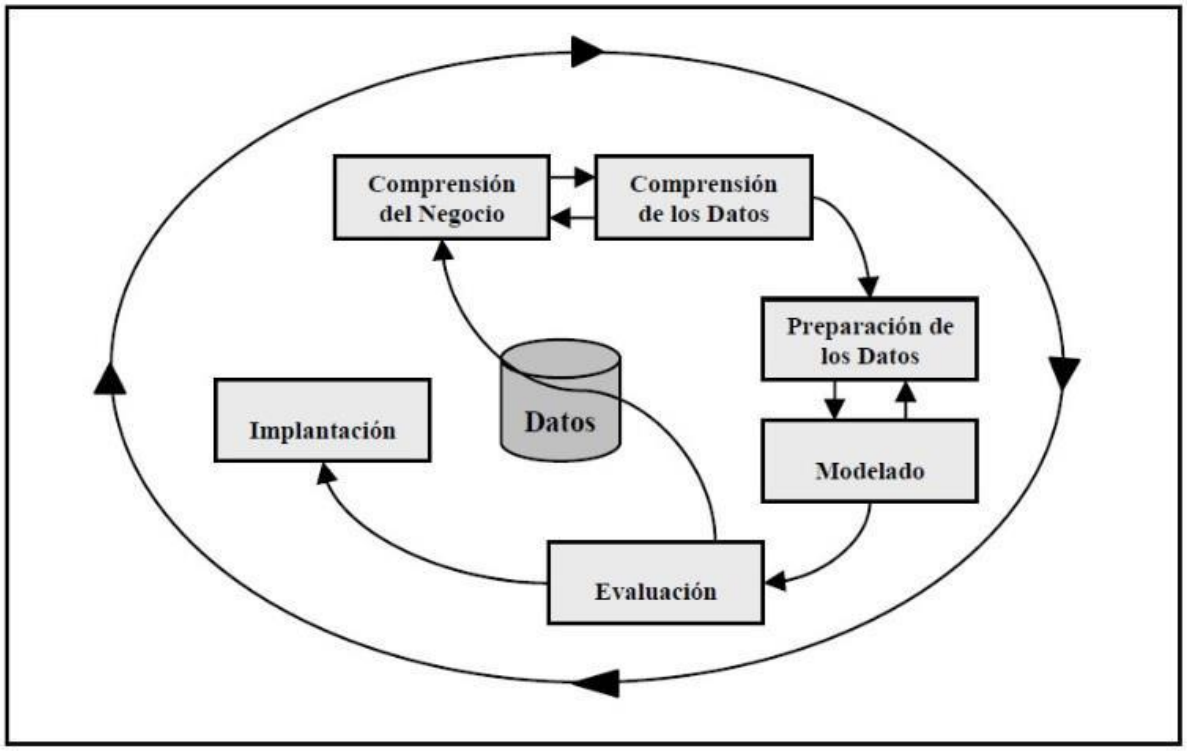


Fig. 3 Modelo de proceso CRISP– DM (Chapman, Clinton, Kerber, Khabaza, Reinartz, Shearer and Wirth 2000)

La secuencia en que se desarrolla cada una de estas fases no es necesariamente rígida, en cada fase se plantean una serie de subtareas, es decir, las tareas generales se proyectan a tareas específicas, pero en ningún momento se plantea como darles solución.

Metodología de John Durkin

Metodología propuesta por John Durkin "Metodología de ingeniería del conocimiento" una de las facilidades que brinda es la poca documentación que genera y que la misma está pensada para un equipo de desarrollo pequeño, consta de seis fases para su desarrollo las cuales aparecen bien detalladas (Durkin and Durkin 1998).

Aunque las metodologías antes mencionadas fueron creadas para guiar el trabajo en el desarrollo de un SE, el éxito del producto depende de cuan bien se apliquen las fases y tareas que se plantean en cada una de las metodologías.

Por las características que presenta la investigación es importante resaltar que no se aplicarán todas las fases ni tareas que propone la metodología seleccionada. A continuación, se describen las fases y tareas que se ejecutarán para llevar a cabo la construcción de ProFeatPred.

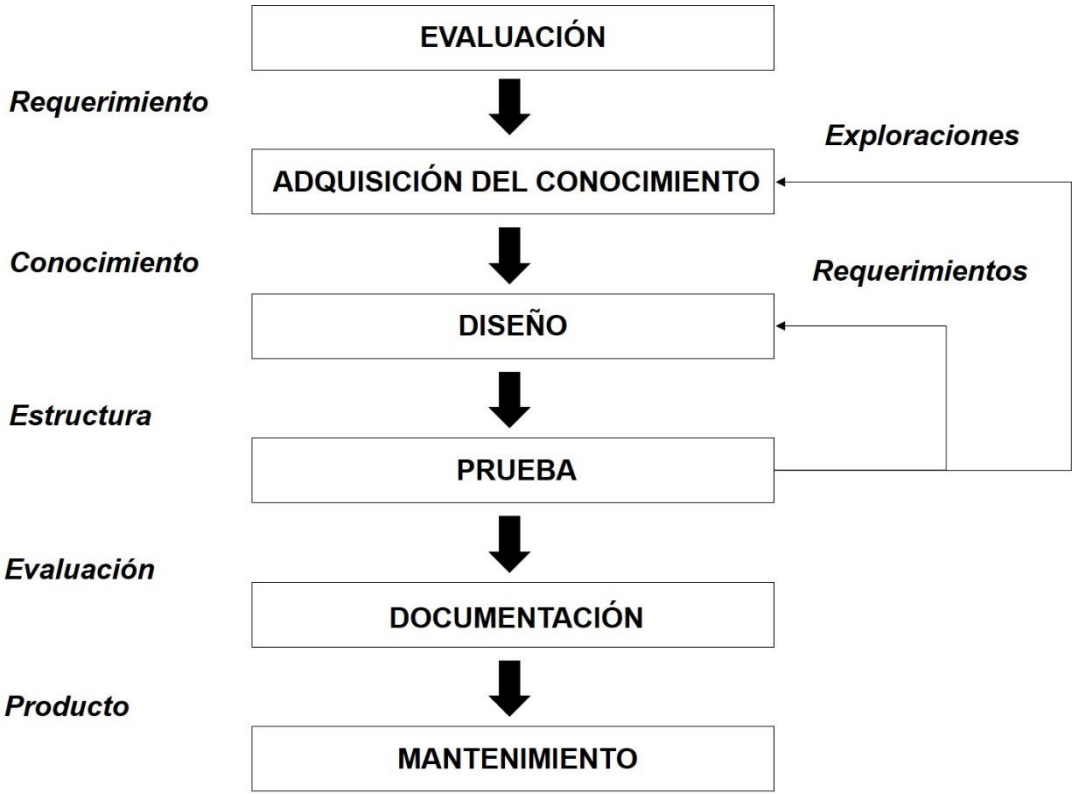


Fig. 4 Fases y tareas que se ejecutarán para llevar a cabo la construcción de ProFeatPred

Fase 1: Evaluación

Tarea 1: Determinar Motivación para el Esfuerzo

En esta tarea es primordial darle respuesta a la siguiente interrogante ¿Por qué está la organización motivada para seguir un SE? En algunas entidades se desea dar solución a un problema particular mientras que a otras las motiva el interés por ver qué puede hacer la tecnología por ellos. Por lo que pueden asumir dos posiciones al incursionar en la tecnología de SE, conducida por el problema, cuando el problema ya se ha identificado, conducido por la solución, por un interés general o curiosidad hacia la tecnología.

Tarea 2: Estudio de Viabilidad

Lo primero en esta tarea es determinar si el proyecto tendrá éxito. Para eso es necesario considerar dos puntos:

Primero: Una lista de activos es verificada en el proyecto. Estos activos incluyen los recursos propios, recurso de conocimiento y personal del proyecto.

- ✓ Disponibilidad de conocimiento para la solución del problema.
- ✓ Disponibilidad de un ingeniero del conocimiento.

- ✓ Disponibilidad del software de desarrollo del sistema.
- ✓ Disponibilidad de facilidades del computador.

Segundo: Aun cuando un proyecto reúne los requerimientos verificados hay otros asuntos que pueden dañar el éxito completo de un proyecto. Un sistema experto puede fallar por razones que caen dentro de las tres categorías: problema, gente y despliegue.

Tarea 3: Escribir el proyecto propuesto

Es importante escribir una propuesta del proyecto que documente los esfuerzos esperados. Esta propuesta deberá documentar por qué el proyecto es importante, y cómo se ejecutará el esfuerzo. La discusión de cada uno de esto puntos, debe ser breve y puntual.

Fase 2: Adquisición del conocimiento

La adquisición del conocimiento es un proceso cíclico que se ve reflejado durante todo el desarrollo de la aplicación. La recolección del conocimiento, su interpretación y análisis, y el diseño de métodos para recolectar conocimiento adicional son tareas primordiales en esta fase.

La recolección del conocimiento: se hace a través de técnicas de recopilación de información principalmente la tormenta de ideas que se emplea con el objetivo de identificar características y capacidades con las que debe cumplir ProFeatPred teniendo en cuenta los criterios del tutor de la investigación y los autores, además de la documentación existente de estudios anteriores relacionados con el tema abordado.

La interpretación: toda la información recolectada envuelve la identificación de piezas clave de conocimiento, como conceptos, reglas y estrategias lo cual es necesario documentar e interpretar para su uso posterior.

El análisis: abarca el estudio de las piezas clave del conocimiento las cuales fueron seleccionadas durante la tarea de interpretación. Esto brinda una visión de las teorías en la organización del conocimiento y estrategias de solución de problemas.

El diseño: siguiendo la realización de las tareas anteriores, se forma una nueva comprensión del problema. Este esfuerzo puede haber expuesto nuevos conceptos que necesitan exploración extensa.

Fase 3: Diseño

Esta fase comienza con la selección de la técnica de representación del conocimiento y la estrategia de control. Sigue con la selección de una herramienta de software que reúne mejor las necesidades del problema. Un sistema prototipo pequeño es más tarde construido para validar el proyecto y proporcionar

una guía para el trabajo futuro. El sistema es entonces extensamente desarrollado y refinado para encontrar los objetivos del proyecto. Este proceso es estructurado de acuerdo a las siguientes tareas:

Tarea 1: Técnica de Representación del Conocimiento

Tarea 2: Técnica de Control

Tarea 3: Desarrollo de Prototipo

Tarea 4: Desarrollo de Interfaces

Tarea 5: Desarrollo del Producto

Tarea 1: Técnica de Representación del Conocimiento

Se debe de escoger la técnica que mejor muestre la manera en que el experto modela el conocimiento del problema mentalmente, se debe además considerar los recursos y capacidades de la organización.

Tarea 2: Técnica de Control

Se seleccionan las técnicas de control, las cuáles pueden ser encadenamiento hacia delante o encadenamiento hacia atrás, la selección de una de estas depende de la existencia o no de una hipótesis a demostrar.

Tarea 3: Desarrollo de Prototipo

Por lo general los proyectos de sistemas expertos comienzan su desarrollo construyendo un prototipo, que no es más que un modelo donde queda reflejada la estructura básica, que representa y procesa el conocimiento del problema. Aunque el prototipo es sólo una pequeña versión del sistema final en estos se logra una aproximación del sistema experto, confirma las técnicas de representación de conocimiento y estrategias de control, además de proporcionar una vía de adquisición de información.

Tarea 4: Desarrollo de Interfaces

Se definen las interfaces con las que el usuario interactuará. El desarrollo de estas debe empezar con el desarrollo del prototipo del sistema experto. Las estrategias para alcanzar un diseño eficaz son la consistencia, la claridad, el control y colores de la pantalla.

Tarea 5: Desarrollo del Producto

Con el desarrollo del prototipo, se sostienen las sesiones de extracción de conocimiento y se realizan las pruebas. Logrando refinar la capacidad del sistema. En un modo evolutivo, el prototipo del sistema empieza a asumir la forma del sistema final.

Fase 4: Prueba

Al concluir el desarrollo de ProFeatPred se realizarán pruebas utilizando conjuntos de proteínas no usadas en el proceso de desarrollo del SE.

Fase 5: Documentación

Después de llevar cierto tiempo de trabajo se debe encontrar una cantidad de información abrumadora. Para manejar esta situación, se debe decidir temprano sobre algún método para documentar efectivamente esta información. Dado a que a menudo es necesario consultar información recopilada para añadir nueva información o estudiar la ya descubierta se da la necesidad de documentar el conocimiento, los gráficos de conocimiento, el código fuente, las pruebas y los reportes.

La documentación debe ser organizada para facilitar el desarrollo del sistema y la escritura de los reportes. Para muchos proyectos de sistema expertos es necesario escribir un reporte final. El contenido del reporte final del proyecto debe incluir lo siguiente, página de título, tabla de contenidos, resumen, visión global del proyecto, revisión bibliográfica, descripción de la aplicación, resultado de las pruebas, referencias, bibliografía y anexos.

1.3. Sistemas Expertos en la bioinformática

Los Sistemas Expertos han sido diseñados para facilitar las tareas en múltiples campos de aplicación, no están diseñados para reemplazar al experto humano sino para ayudar en la toma de decisiones y proporcionar equivalentes resultados, estos se aplican a una gran diversidad de campos y/o áreas, por ejemplo: Militar, Informática, Telecomunicaciones, Química, Derecho, Aeronáutica, Geología, Arqueología, Agricultura, Electrónica, Transporte, Educación, Medicina, Finanzas y Gestión (Badaró, Ibañez and Agüero 2013).

En la bioinformática los SE tienen gran utilidad, ya que en la actualidad es uno de los campos de la ciencia más dinámicos y con más proyección. “La bioinformática es la ciencia del uso de la información para entender la biología. Hablando estrictamente, la bioinformática es un subconjunto del campo mayor de la biología computacional, siendo esta última la aplicación de técnicas analíticas cuantitativas en el modelado de sistemas biológicos (Gibas and Jambeck 2001). A continuación, se muestran ejemplos de SE utilizados en la bioinformática que predicen propiedades biológicas de interés.

FoldRate: A Web-Server for Predicting Protein Folding Rates from Primary Sequence

Es un servidor web que predice rápidamente la tasa de plegamiento de una proteína a partir de su secuencia de aminoácidos. En este sistema se fusionan tres predictores individuales los cuales utilizan como información de entrada las características: tamaño efectivo de plegamiento, α-hélice efecto y hoja-β efecto, respectivamente (Chou and Shen 2009).

D2N: Distance to the native

D2N (distancia al nativo) es un sistema experto que estima la desviación cuadrática media de cualquier estructura sin previo conocimiento de la estructura nativa. D2N diseña tres modelos de entrenamiento basados en RMSD, que se combinan y predicen la distancia de cualquier estructura dada a su estado nativo. Se utiliza la técnica de aprendizaje automático Random Forest para regresión, el cual genera múltiples árboles, permitiendo lograr una alta precisión y además desarrolla modelos de aprendizaje para predecir las puntuaciones Template Modeling (Steinbeck et al.) y Global Distance Test (GDT) (Mishra et al. 2014).

Expert QSAR system for predicting the bioconcentration factor under the REACH regulation

Este sistema predice el factor de bioconcentración. Identifica y combina el mejor modelo Bioconcentration Factor (BCF) para cada clase entre tres modelos VEGA y una ecuación basada en octanol-water (KOW). La clase se predijo utilizando la clasificación por consenso, combina dos árboles de clasificación: el primero identifica compuestos subestimados de KOW, mientras que el segundo identifica compuestos que están sobrestimados por KOW. Cada modelo tiene su propio dominio de aplicabilidad (Badaró, Ibañez and Agüero), calculado sobre la base de los descriptores moleculares gama (Grisoni et al. 2016).

INPS: predicting the impact of non-synonymous variations on protein stability from sequence

Es una herramienta adecuada para calcular el efecto de polimorfismos en proteínas, cuando la estructura de las mismas no está disponible. El INPS se basa en una regresión de soporte vectorial (SVR) implementada por el paquete libsvm y utiliza como alternativa (el denominado puntaje 'HMM'), la información evolutiva fue codificada por medio de un modelo HMM obtenido ejecutando el programa hmmbuild de la suite HMMER (Fariselli et al. 2015).

ProtDCal aplicación 3: Desarrollo de modelos de identificación de enzimas mediante nuevos descriptores 3D de proteínas

ProtDCal propone la obtención de un nuevo modelo de predicción de actividad enzimática, que se compare favorablemente con métodos que utilicen descriptores libres de alineamiento en la modelación de esta actividad, concentrándose la mayor parte de los métodos en el uso de descriptores derivados de grafos moleculares. En las modelaciones de ProtDCal se utiliza como técnica las máquinas de soporte vectorial (SVM) (Glez et al. 2016).

PRORATE: prediction of protein folding rates from structural topology and complex network properties

PRORATE es un sistema experto que predice la tasa de plegamiento de proteínas en las topologías estructurales y las redes de propiedades complejas basadas en la técnica Regresión por Soporte Vectorial (SVR). Se combinaron ocho parámetros estructurales de topología con diez propiedades de redes complejas, con relación a la red de contacto de proteína (PCN) y la red de interacción de largo alcance (Booch et al.).

Como se evidencia anteriormente, en el núcleo de los SE usados en la bioinformática es común encontrar modelos de aprendizaje y/o estadísticos. Por lo tanto, se decide emplear este tipo de SE en el desarrollo del presente trabajo.

1.4. Modelos de aprendizaje automático y estadístico utilizados

Random Forest

Random Forest (RF) es una técnica de clasificación, la cual consta de una combinación (ensamblado) de árboles predictores {h(**x**,Θ_k), *k=1*, ...}, donde (Θ_k) son vectores aleatorios independientemente equidistribuidos. Cada árbol realiza el voto unitario a favor de la clase más frecuente de la entrada **x**. Para realizar la predicción de un nuevo caso, este realiza un recorrido hacia los nodos terminales de cada árbol. Luego se le asigna la etiqueta (clase) correspondiente al nodo terminal. Este proceso se repite en cada uno de los árboles del ensamblado, y la clase que obtenga el voto mayoritario es reportada como la predicción (Breiman).

K-vecinos más Cercanos

Los K-vecinos más cercanos (K-NN) es una técnica de aprendizaje automático basada en instancias (casos). En este grupo de técnicas las instancias del conjunto de entrenamiento son almacenadas y se emplea una función de distancia para determinar qué caso(s) se encuentra(n) más *próximo*(s) al caso que se pretende clasificar. Resulta importante señalar que el parámetro *k* determina el número de vecinos. Frecuentemente, más de un vecino es usado (k>1) para realizar la clasificación, en estas situaciones la clase mayoritaria de los k-vecinos más cercanos (o la distancia ponderada promedio, si la variable respuesta es numérica) es asignada al nuevo caso (Witten and Frank 2005).

Perceptrón Multicapa

El Perceptrón Multicapa (MLP) es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón simple

(Haykin and Simon). El MLP puede ser totalmente o localmente conectado; en el primer caso, cada salida de una neurona de la capa *i* es entrada de todas las neuronas de la capa *i+1*, mientras que en el segundo cada neurona de la capa *i* es entrada de una serie de neuronas (región) de la capa *i+1* (Haykin and Simon).

Regresión Lineal Múltiple

La Regresión Lineal Múltiple (RLM) es una técnica estadística utilizada para estudiar las relaciones entre una única variable dependiente u objetivo y varias variables independientes (predictores) (Hair et al. 1999). Este modelo puede ser expresado como: $Y=a+b_1X_1+b_2X_2+\ldots b_nX_n$, donde, Y es la variable dependiente o explicada, a es la intersección o término constante, X_1; X_2; X_n son las variables independientes o explicativas, y b_1; b_2;…; b_n son los parámetros que miden la influencia que las variables explicativas (X_i) tienen sobre la variable objetivo (Y) (Hair, Anderson, Tatham and Black 1999)

1.5. Descriptores de contacto

Los descriptores son números que se obtienen a partir de la aplicación de un algoritmo sobre una representación simbólica de una molécula. Estos números tienen amplia utilidad pues constituyen la base para la obtención de modelos para predecir determinadas actividades y/o propiedades químicas de las moléculas. En este epígrafe se aborda un tipo de descriptores denominados de *contacto*, los cuales se definen en el marco de las formas algebraicas *bilineales, lineales y cuadráticas*. Estos parámetros serán aplicados en la predicción de dos propiedades de interés en proteínas: la clase estructural y la velocidad de plegamiento (ver Capítulo 2).

1.5.1. Matrices de contacto

Una de las formas de representación de las proteínas consiste en el empleo de mapas de contacto (Fig.5). Estos están constituidos por una matriz cuadrada y simétrica de ***NxN***, donde ***N*** es la cantidad de aminoácidos en la secuencia. Las filas y las columnas representan los aminoácidos (o residuos) presentes en la secuencia. Las entradas (c_{ij}) de la matriz representan información sobre la interacción (contacto) entre los aminoácidos *i* y *j*. Formalmente, la matriz de contacto (C) se define como sigue:

$$c_{ij} = \begin{cases} 1 \text{ si } \mathrm{d}_{\min} \le d_{ij} \le \mathrm{d}_{\max} \\ 0 \text{ en otro caso} \end{cases}$$

donde, d_{min} y d_{max} son las distancias mínimas y máximas y se definen por el investigador y d_{ij} es la distancia euclidiana entre los átomos de carbono-alfa de los aminoácidos *i* y *j*. Cuando no se realizan transformaciones sobre la matriz de contacto, esta se denomina matriz no estocástica ($_{ns}$C). Además de la

matriz $_{ns}C$ se emplean las matrices de contacto simple-estocástica ($_{ss}C$) y de probabilidad mutua ($_{mp}C$). Para detalles en la obtención de estas matrices ver (Torres 2016).

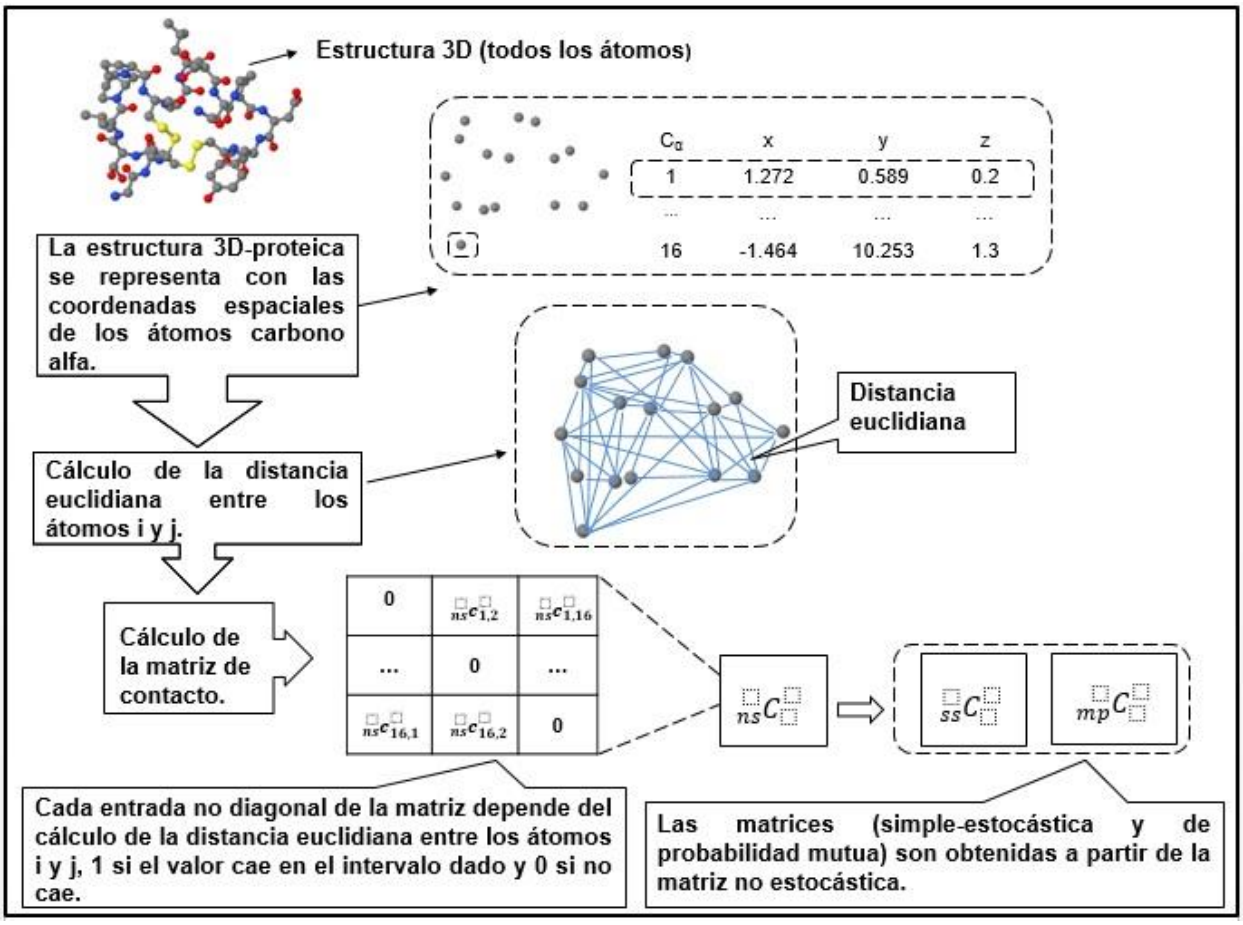


Fig. 5 Representación de proteínas a través de mapas de contacto

1.5.2. Definición matemática de los descriptores de contacto totales y locales

Si una proteína está compuesta por *n* aminoácidos, entonces los *índices de contacto bilineales, cuadráticos y lineales* para el aminoácido *"a"* se calculan como formas bilineales, cuadráticas y lineales, respectivamente, en la base canónica de $\mathbb{R}^n$:

$$ {}_bL_a = b^a(\bar{x},\bar{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^a x^i y^j = [X]^T\, \mathrm{C}^a [Y] \quad \forall\, a = 1,2,\dots,n \qquad (\mathbf{1}) $$

$$ {}_qL_a = q^a(\bar{x},\bar{x}) = q^a(\bar{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^a x^i x^j = [X]^T\, \mathrm{C}^a [X] \quad \forall\, a = 1,2,\dots,n \qquad (\mathbf{2}) $$

$$ {}_fL_a = f^a(\bar{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^a u^i x^j = [U]^T \, \mathrm{C}^a [X] \quad \forall\, a = 1,2,\ldots,n \qquad (\mathbf{3}) $$

donde, (x_m^i, y_m^j) son las componentes de los vectores macromoleculares $(\bar{x}_m, \bar{y}_m)$ en la base canónica de $\mathbb{R}^n$. $[X]$ e $[Y]$ son vectores columnas (matrices de nx1) de las coordenadas de los vectores $\bar{x}_m, \bar{y}_m$, respectivamente; $[X]^T$es la transpuesta (matriz de 1xn) del vector de propiedades $[X]$ y $[U]$ es el vector unidad. Los coeficientes C_{ij}^ason los elementos de la *matriz bidimensional de contacto de nivel aminoacídico* (Borroto, Díaz, de la Vega, Grau, Ponce and Monteagudo), los cuales se obtienen a partir de la *matriz bidimensional de contacto total* (proteína como un todo) C como sigue:

$$ \begin{aligned} c_{ij}^a &= c_{ij} \qquad \text{si } i = a \wedge j = a \\ &= \frac{1}{2} c_{ij} \quad \text{si } i = a \vee j = a \\ &= 0 \qquad \text{en otro caso} \end{aligned} \quad \textbf{(4)} $$

Si una proteína es particionada entre "A" aminoácidos, entonces la matriz C puede ser particionada en "A" matrices de nivel aminoacídico C^a y la matriz total C es la suma de todas las matrices de nivel atómico. Por lo tanto, cada matriz C^a determina un índice de nivel aminoacídico [denotado aquí como LOVI acrónimo de Local Vertex Invariant] para el aminoácido *"**a**"* (L_a). De esta forma, los índices totales (proteína como un todo) se calculan como formas bilineales, cuadráticas y lineales y pueden ser representados como un vector $\bar{L}$ de longitud *n*, donde cada entrada corresponde al índice bilineal, cuadrático o lineal para el aminoácido"***a***". Entonces, a partir de la definición anterior, los índices totales se calculan como combinaciones lineales de los índices de nivel aminoacídico. Como se aprecia en las (Ecs. 5, 6 y 7) los índices bilineales, cuadráticos y lineales se definen mediante la sumatoria de las entradas L_a del vector $\bar{L}$, lo cual es equivalente al producto entre el vector de [propiedad $([X]^T)$ o vector unidad ($[U]^T$)], la matriz C y el vector de propiedad $[Y]$.

$$ b \quad (\bar{x}, \bar{y}) = \sum_{a=1}^{n} {}_bL_a = [X]^T C [Y] \quad \forall\, a = 1,2,\ldots,n \qquad (\mathbf{5}) $$

$$ q \quad (\bar{x}) = \sum_{a=1}^{n} {}_qL_a = [X]^T C [X] \quad \forall\, a = 1,2,\ldots,n \qquad (\mathbf{6}) $$

$$ f \quad (\bar{x}) = \sum_{a=1}^{n} {}_fL_a = [U]^T C [X] \quad \forall\, a = 1,2,\ldots,n \qquad (\mathbf{7}) $$

Además de codificar información de una proteína como un todo, los tres enfoques matriciales propuestos ($_{ns}C$, $_{ss}C$, $_{mp}C$) pueden ser utilizados para codificar información de ciertos fragmentos de interés. Por lo tanto, a partir de la matriz bidimensional global C se puede obtener la *matriz bidimensional de fragmento-local*

(C_F). La matriz C_F contiene información sobre la interacción entre aminoácidos pertenecientes a determinados fragmentos polipeptídicos ***F*** y sus elementos c_{ijF} se calculan como sigue:

$c_{ijF} = c_{ij}$, si $(i \wedge j) \in F$

$c_{ijF} = \frac{1}{2} c_{ij}$, si $(i \vee j) \in F$, *pero no ambos* ***(8)***

$c_{ijF} = 0$, en cualquier otro caso

Por lo tanto, los *índices de contacto bilineales, cuadráticos y lineales de fragmento-local* se calculan mediante las expresiones:

$$b_F(\bar{x}, \bar{y}) = \sum_{a=1}^{n} {}_{bF}L_a = [X]^T \mathrm{C}_F [Y] \quad \forall a = 1,2,\dots,n \qquad (\mathbf{9})$$

$$q_F(\bar{x}) = \sum_{a=1}^{n} {}_{qF}L_a = [X]^T \mathrm{C}_F [X] \quad \forall a = 1,2,\dots,n \qquad (\mathbf{10})$$

$$f_F(\bar{x}) = \sum_{a=1}^{n} {}_{fF}L_a = [U]^T \mathrm{C}_F [X] \quad \forall a = 1,2,\dots,n \qquad (\mathbf{11})$$

donde, ${}_{bF}L_a, {}_{qF}L_a, {}_{fF}L_a$ son los índices de *contacto bilineales, cuadráticos y lineales* de *fragmento*-local para el aminoácido *"**a**"*, los cuales se calculan de igual forma a sus análogos totales. El término fragmento (González-Díaz, González-Díaz, Santana, Ubeira and Uriarte) se refiere no solo a secuencias de aminoácidos, donde los aminoácidos "*i"* e "*i+1"*se encuentran unidos por un enlace peptídico, sino también a grupos de aminoácidos que se encuentran distantes en la estructura primaria. En el presente trabajo, los índices locales se calculan utilizando dos tipos de fragmentos: por *tipos de aminoácidos* y por *agrupaciones de aminoácidos*. En el primer caso, los aminoácidos se clasifican de acuerdo a la naturaleza de su cadena lateral, dando lugar a los siguientes fragmentos: apolar (RAP), polares con carga positiva (RPC), polares con carga negativa (RNC), polares no cargados (RPU), aromáticos (Badaró, Ibañez and Agüero)jar y alifáticos (Ruiz-Blanco et al.). También se definen las siguientes agrupaciones de aminoácidos: aminoácidos que no favorecen el plegamiento y/o no se encuentran con frecuencia formando hélices-α u hojas-β (UFG), aminoácidos que favorecen la formación de hélices-α (FAH), aminoácidos que favorecen la formación de hojas-β (FBS) y aminoácidos que favorecen la formación de giros-β (AFT). Finalmente, se definen agrupaciones que contienen los aminoácidos del mismo tipo (R grupo), es decir, 20 fragmentos, uno por cada aminoácido natural.

1.6. Herramientas, tecnologías y lenguajes de programación

Herramientas CASE: Visual Paradigm para UML 8.0

Visual Paradigm para UML es una herramienta para el desarrollo de aplicaciones utilizando modelado UML ideal para la construcción de sistemas a gran escala representando todo tipo de diagramas, brinda confiabilidad y estabilidad en el desarrollo orientado a objetos. Esta herramienta permite aumentar la calidad del software, mejor productividad en el desarrollo y mantenimiento del software. Entre las características fundamentales que determinaron su selección se destaca que: apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes; brinda apoyo adicional en cuanto a generación de artefactos automáticamente (Hernández et al. 2016b).

Entorno de desarrollo: NetBeans 8.1

La plataforma NetBeans logró ser creada con la contribución de la comunidad de código abierto, siendo un paquete IDE bien diseñado que puede ser usado para la creación de cualquier tipo de aplicaciones de escritorio y web. Es un ambiente integrado de desarrollo de código abierto para desarrolladores de software que intentan construir aplicaciones usando mayormente Java, o algún otro lenguaje de programación popular como C++, PHP, entre otros. NetBeans es también una plataforma de ejecución de aplicaciones, es decir, facilita la escritura de aplicaciones Java, proporcionando una serie de servicios comunes, que a su vez están disponibles a través del IDE (Cumbal et al. 2009).

Herramienta para la confección de modelos de aprendizaje: WEKA 3.7.10

Weka 3.7.10 es una herramienta que se utiliza en diferentes áreas, específicamente en la investigación y la docencia, se basa en el modelado de algoritmos implementados en otros lenguajes de programación, contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificación, regresión, *clustering*, asociación y visualización, está orientada a la extensibilidad por lo que añadir nuevas funcionalidades es una tarea sencilla. Además está programado en Java, es independiente de la arquitectura, ya que funciona en cualquier plataforma sobre la que haya una máquina virtual Java disponible (Morate 2008).

Herramienta para el desarrollo de modelos de regresión lineal múltiple: Mobydigs 1.0

MobyDigs es un software que ha sido desarrollado para la selección de un subconjunto de variables en regresión y análisis de clasificación por algoritmos genéticos. Es una aplicación que puede ejecutarse en plataformas Windows de 32 bits y fue implementada en Microsoft Visual Basic 6.0. Es un software que

extiende de estrategias de la genética, basado en la evolución del modelo de una sola población a un modelo genético más complejo. Estas poblaciones evolucionan independientemente una de otras, y después de un número de iteraciones, pueden ser combinadas de acuerdo a diferentes criterios, obteniendo así una nueva población con diferentes capacidades evolutivas (Todeschini et al. 2003).

Biblioteca Chemistry Development Kit (CDK) 1.4.19

Es una biblioteca de estructuras de datos y algoritmos útiles para el trabajo con diferentes formatos químicos. Está destinada para el uso de programadores que desean ahorrar algo de esfuerzo mediante la reutilización de código integrándose así en diversos entornos, obteniendo un mejor funcionamiento. CDK es de código abierto, implementada en Java para problemas relacionados con la informática, química y la bioinformática, el diseño de clases que presenta está relacionado con estas materias, lo cual la convierte en una herramienta fácil de entender por especialistas en ciencias de la vida, y fácil de utilizar por especialistas en informática con conocimientos básicos en estas ciencias. Es ampliamente utilizada por varios proyectos entre los que se destacan CDK-Taverna, Sinfony1s, Bioeclipse2, entre otros (Steinbeck, Han, Kuhn, Horlacher, Luttmann and Willighagen 2003).

Lenguaje de programación: Java 1.8

El lenguaje de programación Java es un lenguaje orientado a objetos y soporta las tres características propias del paradigma orientado a objetos: encapsulación, herencia y polimorfismo. Puede ejecutar aplicaciones en cualquier plataforma donde esté instalada la Máquina Virtual de Java, pues el código que genera no es específico de una plataforma en particular. Además de las características mencionadas con anterioridad, este lenguaje es simple, distribuido, robusto, seguro, de altas prestaciones, multitarea y dinámico (Deitel 2004).

Herramienta para reducir la dimensionalidad: IMMAN 1.0

IMMAN es un software multiplataforma, desarrollado en el lenguaje de programación Java, comprende dos clases principales que manejan los conjuntos de datos sobre los cuales se van a realizar los cálculos, la clase DataSet y la clase Variable. Los formatos de archivo de entrada aceptados por IMMAN son archivos de valores separados por tabulación y comas (.txt, .csv). Realiza cálculos en tiempo real de múltiples archivos de conjuntos de datos, posee funciones de preprocesamiento de datos, como la sustitución de valores perdidos o eliminación de características, selección de datos, exploración o inspección de datos y partición de conjuntos de datos basados en funciones. Además proporciona la posibilidad de realizar un ranking de un solo parámetro o conjunto (multi-criterio) ranking (Urias et al. 2015).

Conclusiones de capítulo

Después de haber realizado la fundamentación teórica del módulo descriptores de contacto del software ToMoCoMD-CAMPS y del sistema experto ProFeatPred, se arribaron a las siguientes conclusiones:

- ✓ Se alcanzó un mejor dominio sobre los conceptos de la inteligencia artificial y los sistemas expertos, y de la arquitectura de estos.
- ✓ Para la creación de ProFeatPred se seleccionó la metodología de John Durkin.
- ✓ Se mostraron ejemplos de sistemas expertos utilizados en la bioinformática, en los cuales se aprecia que es común encontrar modelos de aprendizaje automático y/o estadísticos.
- ✓ Se definieron matemáticamente los descriptores de contacto.
- ✓ Como herramientas se propone la utilización del lenguaje de programación el Java en su versión 1.8 y el entorno de desarrollo NetBeans 8.1. Además se determina emplear la herramienta WEKA 3.7.10 para el desarrollo e integración al SE de los modelos de clasificación. Para el modelado se utiliza Visual Paradigm para UML en su versión 8.0, para la confección de los modelos de regresión se trabaja con el Mobydigs en su versión 1.0 y para la manipulación del formato químico PDB se emplea Biblioteca Chemistry Development Kit (CDK) 1.4.19.

Capítulo 2: Sistema Experto para la Predicción de Propiedades Biológicas en Proteínas

En este capítulo se realiza el análisis y diseño del módulo descriptores de contacto; el modelo de dominio, los requisitos funcionales y no funcionales, los diagramas de casos de uso y de clases del sistema, conjuntamente se describen los patrones usados. De manera precisa se explica cómo se lleva a cabo la fase de evaluación, adquisición del conocimiento y diseño de la metodología estudiada. Además, se aborda el proceso de desarrollo de los modelos de clasificación y de regresión.

2.1. Análisis y diseño del módulo descriptores de contacto

2.1.1. Modelo de dominio

El modelo de dominio aborda todos los temas relacionados con un problema específico. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el dominio del problema (Kendall 2005). A continuación, se muestra el modelo de dominio del módulo descriptores de contacto Fig.6 y una breve descripción de los conceptos que intervienen en él.

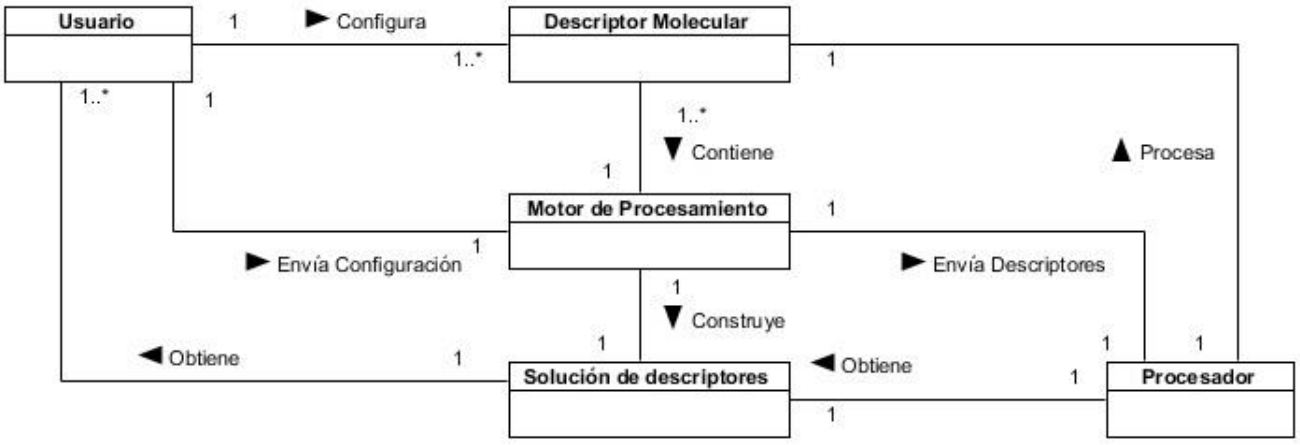


Fig. 6 Modelo de dominio del módulo descriptores de contacto

- ✓ *Usuario:* Representa la entidad que configura un Descriptor Molecular y desea procesarla.
- ✓ *Descriptor Molecular:* Representa la entidad que es configurada por un determinado usuario y que se va a procesar.
- ✓ *Motor de Procesamiento:* Representa la entidad encargada de recibir un Descriptor Molecular, enviarlo al Procesador y construir la Solución de Descriptores.
- ✓ *Procesador:* Es la entidad encargada de procesar el Descriptor Molecular y obtener una Solución de Descriptores.

- ✓ *Solución de Descriptores:* Es la entidad final que es enviada al Usuario, obtenida a partir del Procesador.

2.1.2. Requisitos del módulo descriptores de contacto

Los requerimientos de un sistema definen qué es lo que este debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Un requerimiento es una característica de diseño, una propiedad o un comportamiento de un sistema. Estos constituyen la descripción de los deseos o de las necesidades de un producto. Se pueden clasificar en requerimientos funcionales y no funcionales (Sedeño 2013).

Requisitos funcionales

Son declaraciones de los servicios que proveerá el sistema, la manera en que éste reaccionará en situaciones particulares (Sommerville 2005). A continuación, se listan los requisitos funcionales del módulo descriptores de contacto.

RF1. Seleccionar formas algebraicas
RF2. Seleccionar formas matriciales
RF3. Seleccionar operadores de agregación
RF4. Seleccionar propiedades de la cadena lateral de aminoácidos
RF5. Seleccionar los locales
RF6. Crear descriptores a partir de la configuración
RF7. Leer proteínas a partir del formato químico PDB
RF8. Calcular vectores macromoleculares
RF9. Guardar los resultados en formato CSV
RF10. Guardar configuración de descriptores de contacto
RF11. Cargar configuración previamente guardadas de descriptores moleculares

Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo, estándares, etc. (Sommerville 2005). Existen varias categorías para clasificar a los requerimientos no funcionales, siendo las siguientes, las más representativas.

- ✓ Portabilidad

RNF1. El software ToMoCoMD-CAMPS en el cual está implementado el Módulo Descriptores de Contacto es multiplataforma por lo que puede ser utilizado en cualquier sistema operativo.

✓ Usabilidad

RNF2. El sistema debe de contar en la interfaz principal con un menú que indique las opciones que brinda el mismo.

RNF3. Los iconos presentes en el software deben de estar etiquetados para que el usuario se oriente.

✓ Software

RNF4. En la computadora que se desee ejecutar el software deberá contar con una Máquina Virtual de Java en su versión 1.7 o superior.

✓ Hardware

RNF5. Para el uso del software ToMoCoMD-CAMPS se sugiere como requerimientos mínimos un micro-procesador Intel (R) Core-2Duo™ y 2GB de RAM o uno de prestaciones similares.

2.1.3. Modelo de caso de uso

Un modelo de caso de uso describe qué hace un sistema sin explicar cómo lo hace. El modelo de caso de uso presenta al sistema desde la perspectiva de un usuario fuera del mismo (por ejemplo, los requerimientos del sistema) (Kendall 2005).

Actores del sistema

El actor es una abstracción de las entidades externas a un sistema, subsistemas o clases que interactúan directamente con el sistema. Un actor participa en un caso de uso o conjunto coherente de casos de uso para llevar a cabo un propósito global (Hernández et al. 2016a).

En el caso particular de esta investigación quien realiza las invocaciones a los casos de uso del sistema será el cliente como se muestra en la Tabla 1.

Tabla 1. Actor del sistema

Actor	Descripción
Cliente	Es el responsable de configurar un Descriptor Molecular y enviarlo al Motor de Procesamiento para ser computado.

Diagrama de caso de uso

Los diagramas de casos de uso identifican a todos los actores en el dominio del problema. Se utilizan para mostrar el alcance de un sistema, junto con las principales características del mismo y los actores que trabajan con esas características principales (Kendall 2005). Para la realización de esta investigación fueron definidas un conjunto de acciones que deben ser ejecutadas por el actor del sistema. Las interrelaciones

entre las acciones y el actor del sistema son agrupadas en el diagrama de casos de uso del sistema que se muestra en la Fig. 7.

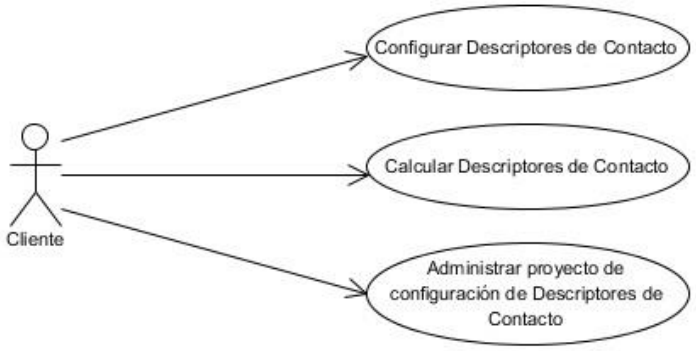


Fig. 7 Diagrama de Caso de uso del sistema

A continuación, en la Tabla 2 se detallan brevemente los casos de usos identificados en el sistema.

Tabla 2. Listado de casos de uso

Orden	Nombre	Prioridad	Descripción
1	Configurar Descriptores de Contacto.	Crítico	El caso de uso comienza cuando un cliente se dispone a configurar un descriptor molecular teniendo en cuenta la propiedad de contacto.
2	Calcular Descriptores de Contacto	Crítico	El caso de uso comienza cuando un cliente ha configurado previamente un descriptor y se dispone a calcularlo.
3	Administrar proyecto de configuración de Descriptores de Contacto.	Crítico	El caso de uso comienza cuando un cliente decide guardar la configuración de un descriptor o cargar una previamente guardada. Por lo antes mencionado se utiliza el patrón de caso de uso CRUD Parcial para agrupar estos requisitos funcionales.

A continuación, en la Tabla 3 se muestra la descripción del caso de uso crítico Calcular Descriptores de Contacto.

Tabla 3. Descripción del caso crítico Calcular Descriptores de Contacto.

Caso de uso 2: Calcular Descriptores de Contacto.

Caso de Uso:	Calcular Descriptores de Contacto
Actores:	Cliente
Resumen:	El caso de uso comienza cuando un cliente ha configurado previamente un descriptor de contacto y se dispone a calcularlo.
Prioridad	Crítico
Flujo Normal de Eventos	
Flujo básico "Calcular Descriptores de Contacto"	
Actor	**Sistema**
1. El caso de uso se inicia cuando el actor se dispone a "Calcular Descriptores de Contacto".	2. El sistema muestra una interfaz con las distintas métricas de contacto (Fig.8)
3. El cliente selecciona la(s) métrica que desea y presiona OK.	
Flujo alterno "Cálculo de la métrica 36"	
Actor	**Sistema**
3a. El actor selecciona la métrica 36.	4. Activa campo para introducir el intervalo deseado.
5. El actor inserta el intervalo deseado y presiona OK.	
5a. El actor no entra un intervalo.	6. Muestra una advertencia que se debe introducir un intervalo para calcular el descriptor (Fig.9).

Prototipo de Interfaz

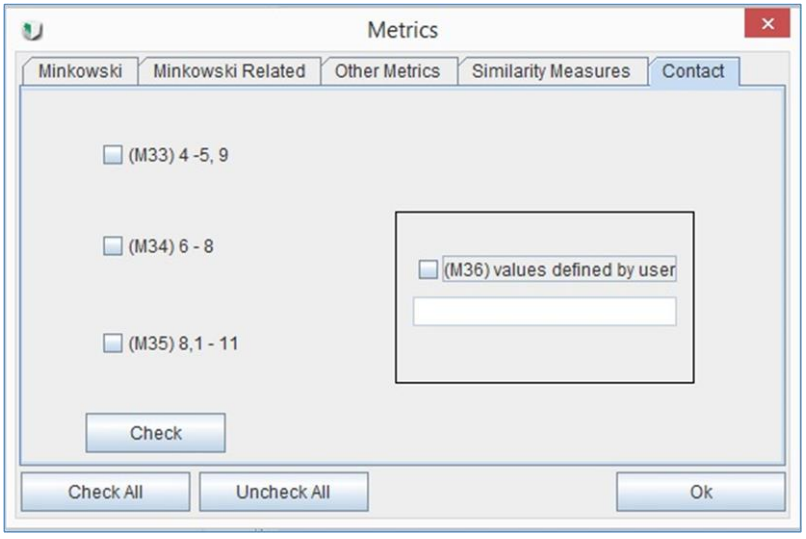


Fig. 8 Interfaz de las distintas métricas de contacto

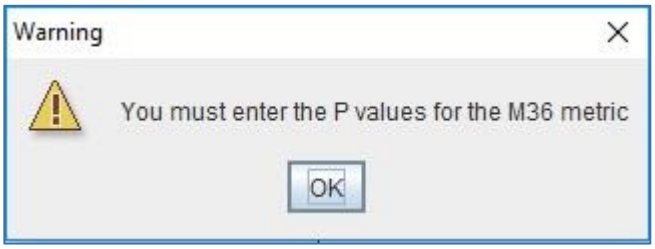


Fig. 9 Advertencia campo vacío

2.1.4. Arquitectura del sistema

La arquitectura de un sistema de información define no solo la estructura y su comportamiento, sino también, el uso, funcionalidad, rendimiento, flexibilidad, reutilización, facilidad de comprensión, restricciones y compromisos económicos y tecnológicos, y por supuesto las vistas de las diferentes etapas que conforman el ciclo de vida del software (Jacobson et al. 2000). Otros autores describen a la arquitectura de un sistema como un conjunto de decisiones de diseño tomadas para un sistema (Tahuiton Mora 2011).

Debido a que el módulo descriptores de contacto está implementado en el software ToMoCoMD-CAMPS, este adopta su estilo arquitectónico, siendo en dos capas ya que presenta una organización jerárquica en las mismas, cada una provee servicios a la capa superior y se alimenta de las prestaciones que brinda la capa inferior. A continuación, en la Fig.10 se evidencia este proceso.

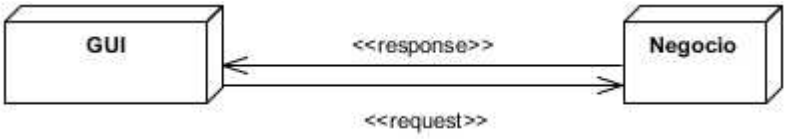


Fig. 10 Arquitectura del módulo descriptores de contacto

En el patrón en dos capas, la capa de presentación, o interfaz gráfica es la que interactúa con el usuario, mostrándole el sistema, proporcionando la posibilidad de configurar los descriptores moleculares a calcular o cargar una configuración previamente guardada. En la capa de lógica de negocio residen las funciones que se ejecutan, se reciben las peticiones del usuario, se procesa la información y se envía la respuesta. Esta capa se comunica con la de presentación, para recibir las solicitudes y presentar los resultados. Aquí se incluyen las clases responsables de calcular los descriptores moleculares.

2.1.5. Diagrama de clases del diseño

Los diagramas de clases muestran las características estáticas del sistema y no representan ningún procesamiento en especial. Un diagrama de clases también muestra la naturaleza de las relaciones entre las clases (Kendall 2005). A continuación en la Fig. 11 se muestra el diagrama de las clases relacionadas

con el módulo descriptores de contacto, teniendo en cuenta que el mismo está implementado en el software ToMoCoMD-CAMPS.

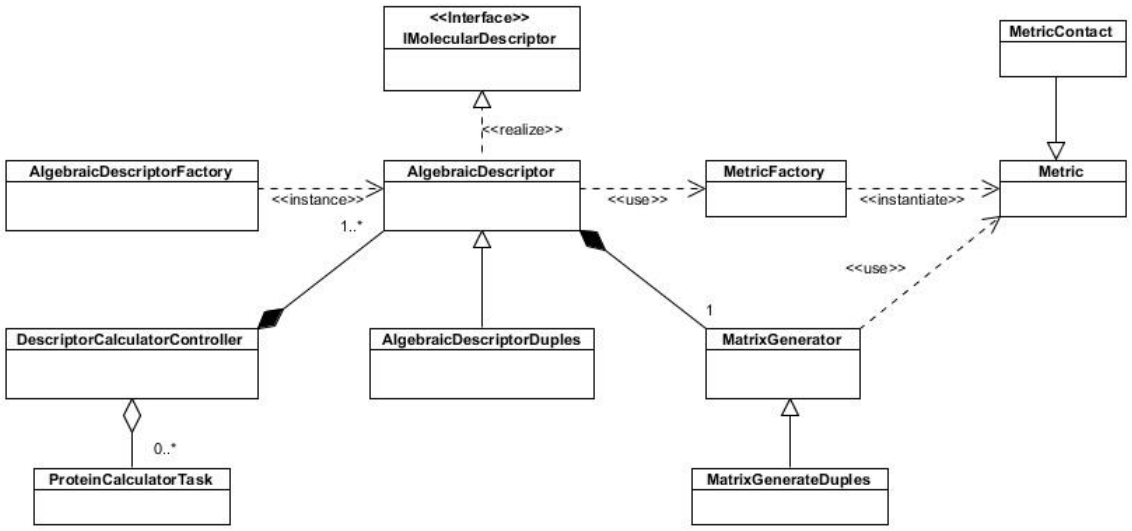


Fig. 11 Diagrama de las clases principales del diseño

Resulta importante señalar que las peticiones realizadas por los usuarios a través de la interfaz de escritorio son procesadas por la biblioteca MuLiMs-MCoMPAs. Este componente está estructurado en paquetes acorde a sus funcionalidades, los cuales se describen a continuación:

- ✓ Las clases *AlgebraicDescriptor*, *AlgebraicDescriptorFactory* y *AlgebraicDescriptorDuples* pertenecen al paquete *tomocomd.camps.mulims.descriptors* el cual se relaciona con el cómputo de los 3D-DMs.
- ✓ Las clases *MatrixGenerator* y *MatrixGeneratorDuples* se encuentran en el paquete *tomocomd.camps.mulims.matrices* el cual construye los enfoques matriciales que se utilizan para representar las interacciones (contactos) entre los aminoácidos de las proteínas.
- ✓ Las clases *ProteinCalculatorTask* y DescriptorCalculatorController pertenece al paquete *tomocomd.camps.mulims.workers* el mismo contiene las clases necesarias para la realización y control del cálculo de los descriptores.
- ✓ Las clases *Metric, MetricContact y MetricFactory* están incluidas en el paquete *tomocomd.camps.mulims.metric* en este se modelan y construyen las métricas para el establecimiento de las interacciones (contactos) entre aminoácidos.

2.1.6. Patrón de diseño

Un patrón de diseño nombra, motiva y explica de forma sistemática un diseño general que afronta un problema de diseño recurrente en los sistemas orientados a objetos. Describe el problema, la solución,

cuándo aplicar la solución y sus consecuencias. También ofrece pistas para la implementación y ejemplos (Vlissides et al. 1995).

El módulo descriptores de contacto se construyó sobre un sistema ya implementado ToMoCoMD-CAMPS por lo que el mismo adopta sus patrones de diseño. Un ejemplo de ello es la utilización del patrón Estrategia y Factoría Abstracta para resolver determinados problemas.

Estrategia

Es un patrón de comportamiento, que define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

Problema: Establecer el contacto entre pares de átomos.

Solución: Se crea una clase *MetricContact* que hereda de *MetricDuplex* y se redefine el método *matrixValue*.

Factoría Abstracta

Es un patrón de creación que proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

Problema: Se necesita independizar la creación de un objeto de tipo *Metric* de la clase que lo utiliza.

Solución: Se implementó una clase *MetricFactory* encargada de crear y retornar un objeto *Metric.*

2.1.7. Diagrama de componentes

Un componente es un grupo de clases que trabajan estrechamente, pueden corresponder código fuente, binario o ejecutable (Torres 2004). El diagrama de la Fig.12 permite modelar la estructura del software y la dependencia entre componentes.

Fig. 12 Diagrama de componentes

- ✓ En el módulo descriptores de contacto existen tres componentes principales*, File.pdb, ToMoCoMD-CAMPS.jar y MuliMs-MCoMPAs.jar.*
- ✓ El componente *File.pdb* es el que contiene la proteína, con todos sus átomos, que serán procesados con el componente ejecutable *ToMoCoMD-CAMPS*.

- ✓ El componente *ToMoCoMD-CAMPS* es un ejecutable que permite cargar los ficheros, configurar los descriptores y guardar la configuración de los descriptores, que se comunica mediante la clase interface *AlgebraicDescriptors* a la biblioteca *ToMoCoMD MuLiMs-MCoMPAs.*
- ✓ El componente *MuliMs-MCoMPAs* es una biblioteca que permite calcular los índices de contacto.

2.1.8. Estándar de codificación

Al comenzar un proyecto de software, se debe establecer un estándar de codificación, el mismo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico (Microsoft 2016). El módulo descriptores de contacto está implementado en el software ToMoCoMD-CAMPS adoptando estándares de codificación del lenguaje Java que se utilizan en dicho sistema.

En Java, las convenciones de nombres para los identificadores se han establecido y propuesto por varias comunidades de Java como Sun Microsystems[1], Netscape[2], AmbySoft[3], etc. Una muestra de las convenciones de nomenclatura establecidas por Sun Microsystems se nombra a continuación, donde un nombre en "CamelCase" es un compuesto de un número de palabras unidas sin espacios, con letra inicial de cada palabra en mayúsculas por ejemplo "CamelCase" (Type-Safe 2014)

- ✓ Los nombres de las clases deben ser sustantivos en *upper CamelCase*, con la primera letra de cada palabra en mayúscula.
- ✓ Los métodos deben ser verbos en *lower CamelCase* o un conjunto de varias palabras que comienza con un verbo en minúsculas; es decir, con la primera letra en minúscula y la primera letra de las palabras siguientes en mayúsculas.
- ✓ Las variables locales, variables de instancia y variables de clase también se escriben en *lower CamelCase*. Los nombres de variable no deben comenzar con los caracteres guión bajo (_) o el signo de dólar ($).
- ✓ Siempre que sea posible inicializar las variables locales donde se declaran.
- ✓ Los nombres de variables deben ser cortos pero significativos.
- ✓ Los nombres de las constantes se deben escribir en mayúsculas separadas por guiones bajos. Los nombres de constantes pueden contener dígitos, pero no como el primer carácter.

[1] "Convenciones de código para el lenguaje de programación Java", Sección 9: "Convenciones de nomenclatura"
[2] "SOFTWARE DE NETSCAPE Normas de codificación GUÍA PARA JAVA", Collab Software Codificación Guía de Normas para Java
[3] "AmbySoft Inc. Estándares de Codificación para v17.01d Java"

A continuación, se presenta un fragmento de código donde se evidencia el uso de los estándares de codificación.

```
public double matrixValue ( IAtom[ ] atoms)
  {
     MetricConfiguration m5 = new MetricConfiguration (MetricType.MINKOWSKI, "m5", new Double [ ]
     {2.00}, false);
     Metric metric = MetricFactory.getMetric (m5);
     double distance = MetricTool.distanceBetweenAtoms (metric, atoms [0], atoms [1]);
     return (distance> = from & & distance < =until ) ? 1:0;
  }
```

2.2. Análisis y diseño del sistema experto

2.2.1. Evaluación

Motivación para el esfuerzo

La motivación para el esfuerzo consiste en determinar por qué está motivada una organización para la realización de un sistema experto, existen dos vías para determinar la motivación para el esfuerzo, una conducida por el problema y otra conducida por la solución (Durkin and Durkin 1998), estas se explican en Metodología de John Durkin en el Capítulo 1. En el caso de ProFeatPred la motivación para el esfuerzo es conducida por el problema que se genera a partir de la necesidad de predecir propiedades biológicas de interés en proteínas.

Estudio de viabilidad

La siguiente tarea en la fase de evaluación es el estudio de la viabilidad que tiene como objetivo fundamental determinar si el proyecto tendrá éxito o no. Para determinar esta tarea se consideran dos puntos a evaluar; el primer punto es que se verifica una lista de activos que debe reunir el proyecto, esta lista se encuentra en el epígrafe Metodología de John Durkin en el Capítulo 1 y se cumple en toda su dimensión; el segundo punto a considerar son asuntos importantes para el éxito del proyecto, los cuales son subjetivos de naturaleza y requieren algún juicio para determinar, ellos incluyen características del problema, características de la gente involucrada del proyecto y asuntos de despliegue. Para ejecutar este punto se utiliza el método de John Durkin que consiste en formar una lista de temas importantes para considerar. Cada tema es luego asignado un peso (entre 0 y 10) que refleja la importancia de cada tema durante la evaluación de un proyecto dado, los números (entre 0 y 10) son atribuidos a cada tema que refleja el grado de creencia en el tema. Este valor es luego multiplicado por el valor del tema para establecer un puntaje por

el tema. Todos los puntajes son luego añadidos y divididos por la suma de los pesos del tema. Este número es limitados entre 0 y 10, y proporciona una estimación de determinación de viabilidad del proyecto (Becerra 2011). Después de haber aplicado el método (Ver Anexo1), el proyecto presenta una viabilidad de 7.78 por lo que se demuestra que es factible realizar el sistema experto.

2.2.2. Adquisición del Conocimiento

Aunque la adquisición del conocimiento es una actividad que involucra toda la ingeniería de conocimiento, desde que se inicia el estudio de viabilidad hasta que finaliza el uso del sistema experto desarrollado, es en esta etapa donde adquiere su mayor uso. Consta de las siguientes tareas: recolección del conocimiento, interpretación, análisis y el diseño de métodos para recolectar conocimiento adicional.

Recolección del conocimiento

En esta tarea se acumula el nuevo conocimiento, es decir, se adquiere el conocimiento del experto. Este esfuerzo solicita la utilización de técnicas de recopilación de información. En el caso de la investigación en curso, la recolección del conocimiento se genera a través de la técnica de tormenta de ideas que surgen entre el tutor y los autores, por lo que requiere tener buenas habilidades de comunicación interpersonal y la destreza para obtener la cooperación del experto. El objetivo principal de esta actividad es definir peculiaridades y propósitos que debe cumplir el sistema ProFeatPred en su desarrollo.

Interpretación del conocimiento

La interpretación de la información anteriormente recolectada consiste en hacer representaciones de la realidad en forma de modelos. Para realizarlos se asignan conceptos, se definen y se tratan de relacionar quedando de esta manera conformado el modelo de dominio que se muestra en la Fig.13, que permite un mejor entendimiento del flujo de trabajo del sistema experto.

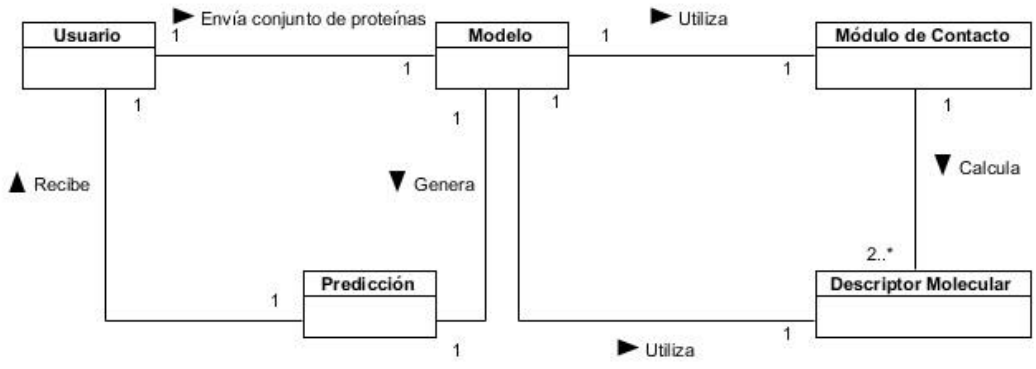


Fig. 13 Modelo de dominio del sistema experto

A continuación se detallan los conceptos representados en el modelo de dominio del sistema experto.

- ✓ *Usuario:* Representa la entidad que suministra un conjunto de proteínas al modelo y recibe una predicción.
- ✓ *Modelo:* Representa la entidad que recibe un conjunto de proteínas descritas en *n* rasgos, las cuales se utilizan para obtener la predicción solicitada por el usuario.
- ✓ *Módulo de Contacto:* Representa la entidad en la cual se calculan los descriptores de contacto.
- ✓ *Descriptor Molecular:* Es la entidad que representa un número obtenido a partir de la aplicación de un algoritmo sobre una representación molecular.
- ✓ *Predicción:* Es la entidad que representa el resultado generado por el modelo.

Análisis del conocimiento

En esta tarea se estructura el conocimiento existente, es decir, se conforman los requerimientos del sistema, describiendo especificaciones con los conocimientos iniciales de los autores, que sirven de base para la realización del sistema, además de activar procesos de representación del sistema a partir de la nueva información.

Requisitos funcionales

RF1. Cargar archivos PDB.

RF2. Predecir clase estructural.

RF3. Predecir velocidad de plegamiento.

RF4. Guardar resultados de la predicción.

Requisitos no funcionales

✓ Usabilidad

RNF1. El sistema debe poseer una interfaz entendible y amigable, de fácil acceso manipulación y que facilite la localización de las diferentes funcionalidades presentes en ella.

✓ Apariencia o interfaz externa

RNF2. El sistema debe poseer una apariencia profesional con un diseño gráfico sencillo proporcionando una mejor interacción entre el usuario y el sistema.

✓ Portabilidad

RNF3. El sistema es multiplataforma por lo que puede ser utilizado en cualquier sistema operativo.

✓ Software

RNF4. Las computadoras que utilizarán el software deben contar con la máquina virtual de Java 1.7 o superior.

✓ Hardware

RNF5. Las computadoras que utilizarán deben tener como requerimientos mínimos un micro-procesador Intel (R) Core-2Duo™ y 2GB de RAM o uno de prestaciones similares.

Diagrama de caso de uso del sistema

El actor juega un papel fundamental, ya que este es el que interactúa con el sistema, en este caso el actor es el usuario que decide cargar PDB, realizar predicciones y guardarlas como se muestra en la Fig. 14.

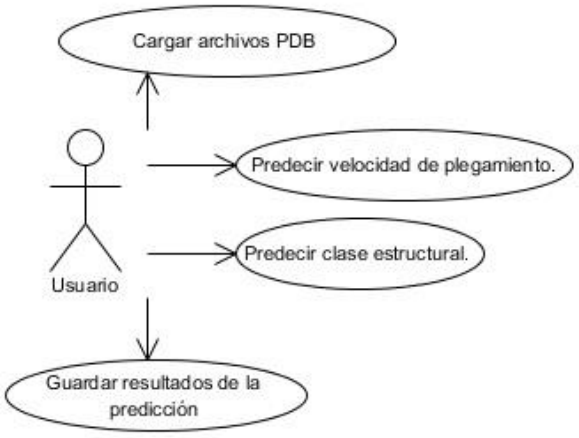


Fig. 14 Diagrama de caso de uso del sistema experto

A continuación en la Tabla 4 se detallan brevemente los casos de usos identificados en el sistema.

Tabla 4. Listado de caso de usos del sistema experto

Orden	Nombre	Prioridad	Descripción
1	Predecir clase estructural.	Crítico	El caso de uso comienza cuando un usuario se dispone a predecir la clase estructural de proteínas.
2	Predecir velocidad de plegamiento.	Crítico	El caso de uso comienza cuando un usuario cliente se dispone a predecir la velocidad de plegamiento.
3	Cargar archivos PDB.	Crítico	El caso de uso comienza cuando un usuario decide cargar archivos PDB.

4	Guardar resultados de la predicción	Crítico	El caso de uso comienza cuando el usuario se dispone a guardar los resultados.

Especificación de los casos de uso del sistema experto

Tabla 5. Especificación de los casos de uso del sistema experto

Caso de uso 1: Cargar archivos PDB	
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario selecciona la opción “Cargar archivos PDB”, la cual consiste en localizar y cargar los archivos con extensión “.pdb” en el sistema.
Complejidad	Baja
Prioridad	Crítico
Flujo de eventos	
Flujo básico “Cargar archivos PDB”	

Actor	**Sistema**
1. El caso de uso inicia cuando el actor selecciona la opción “Cargar archivos PDB”.	2. El sistema muestra una interfaz para localizar y seleccionar los archivos PDB (Fig.15).
3. El actor selecciona los archivos deseados y luego selecciona la opción “Aceptar”.	4. El sistema actualiza la tabla con los nombres de los archivos seleccionados.

Prototipo de Interfaz

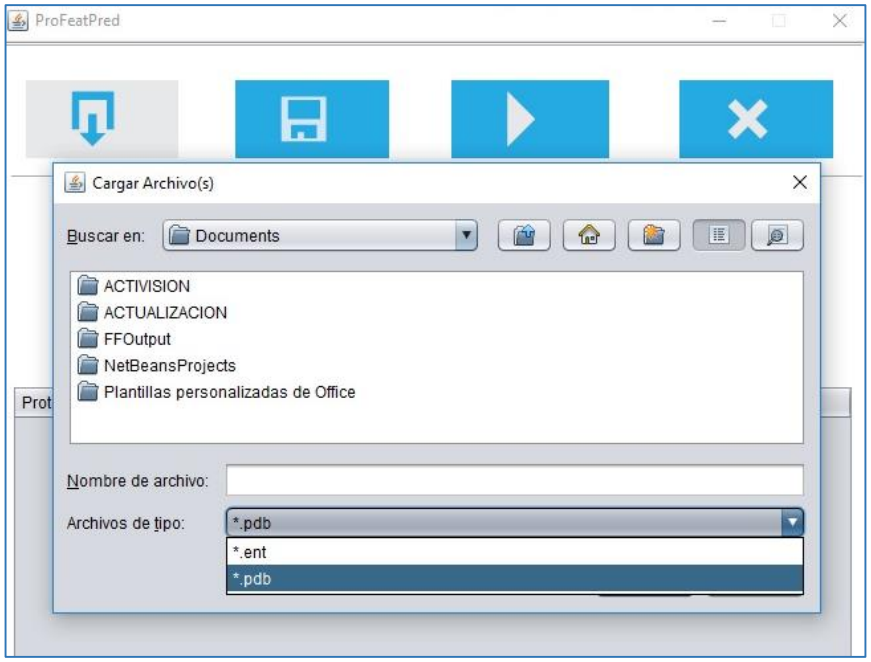


Fig. 15 Cargar Archivo(s)

Flujos alternos "Cancelar"	
Actor	**Sistema**
3a. El usuario selecciona la opción "Cancelar"	El estado del sistema se mantiene sin cambios.
Caso de uso 2: Predecir clase estructural	
Caso de Uso:	Predecir clase estructural
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el actor, selecciona la opción "predecir clase estructural", la cual consiste en determinar la clase estructural para las proteínas cargadas en el sistema. El caso de uso termina cuando se muestra la clase estructural por cada proteína.
Prioridad	Crítico
Flujo Normal de Eventos	
Flujo básico "Predecir clase estructural"	
Actor	**Sistema**
1. El caso de uso se inicia cuando el actor selecciona la opción "Predecir clase estructural".	2. El sistema calcula los descriptores presentes en el modelo de clasificación obtenido con el software WEKA para cada proteína, este modelo predice la clase estructural por cada proteína. 3. Se actualiza en la tabla de resultados la columna correspondiente a la clase estructural (Fig.16).
Flujo alterno "Predecir clase estructural"	
Actor	**Sistema**
	2a. El sistema muestra un listado con los archivos que tienen algún error de formato.

Prototipo de Interfaz

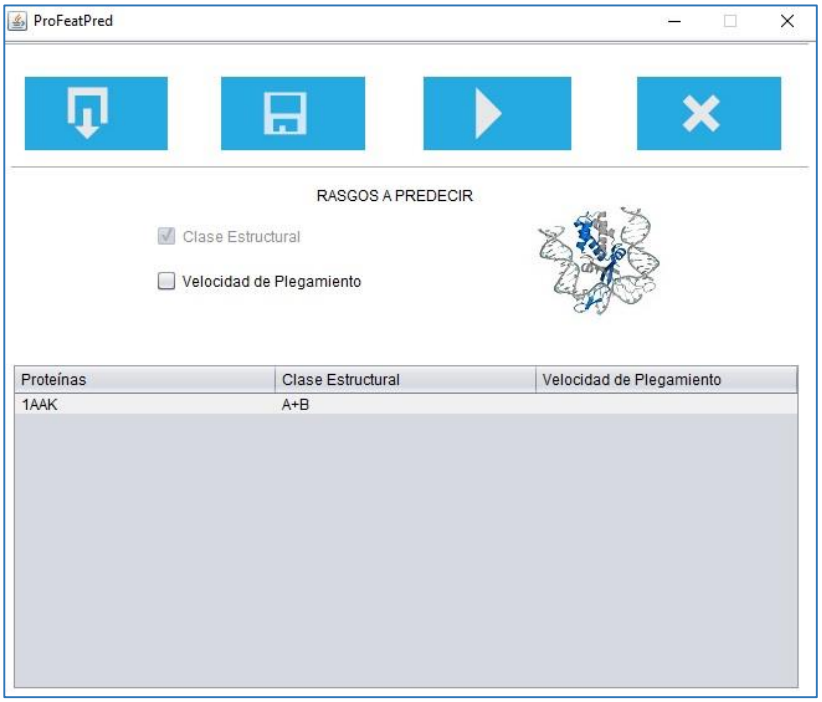


Fig. 16 Predecir Clase Estructural

Caso de uso 3: Predecir velocidad de plegamiento	
Caso de Uso:	Predecir velocidad de plegamiento
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el actor, selecciona la opción “predecir velocidad de plegamiento”, la cual consiste en calcular la velocidad de plegamiento para las proteínas cargadas en el sistema. El caso de uso termina cuando se muestra la velocidad de plegamiento por cada proteína.
Prioridad	Crítico
Flujo Normal de Eventos	
Flujo básico “Predecir velocidad de plegamiento”	
Actor	**Sistema**
4. El caso de uso se inicia cuando el actor selecciona la opción “Predecir velocidad de plegamiento”.	5. El sistema calcula los descriptores presentes en el modelo de Regresión Lineal Múltiple obtenido con el software Mobydigs, este modelo predice la velocidad de plegamiento por cada proteína. 6. Se actualiza en la tabla de resultados la columna

	correspondiente a la velocidad de plegamiento (Fig.17).
Flujo alterno "Predecir velocidad de plegamiento"	
Actor	**Sistema**
	2a. El sistema muestra un listado con los archivos que tienen algún error de formato.

Prototipo de Interfaz

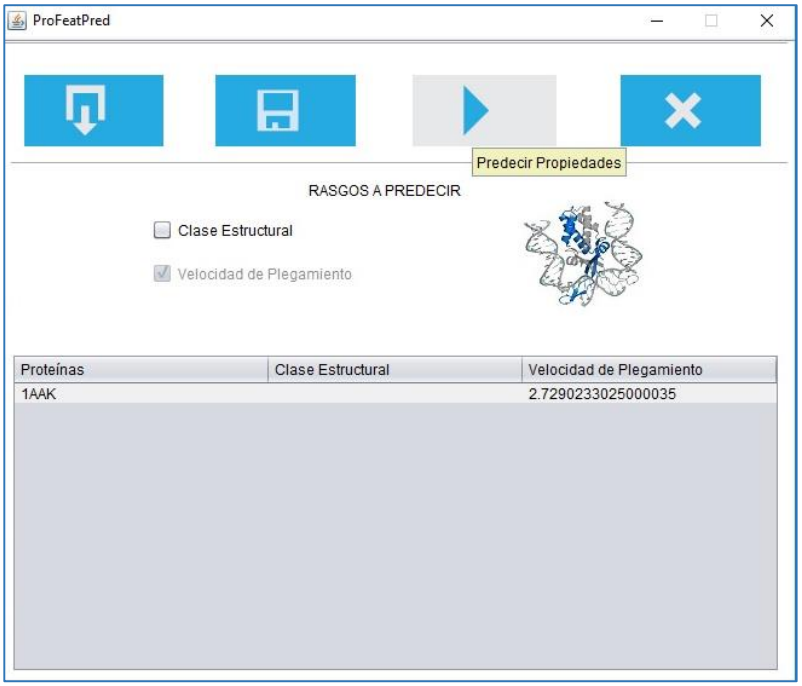


Fig. 17 Predecir Velocidad de Plegamiento

Caso de uso 4: Guardar resultados	
Caso de Uso:	Guardar resultados.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario selecciona la opción "Guardar resultados", la cual consiste en guardar los resultados de las predicciones en un directorio del sistema. El caso de uso termina cuando se guarda un archivo en formato de texto plano que contiene las proteínas y sus respectivas predicciones (Fig.18).
Complejidad	Baja
Prioridad	
Flujo de eventos	

Flujo básico "Guardar resultados"	
Actor	**Sistema**
1. El caso de uso inicia cuando el actor selecciona la opción "Guardar resultados".	2. El sistema muestra una ventana para escoger la carpeta de salida de los resultados.
3. El actor nombra el archivo de salida y selecciona la opción "Aceptar".	4. El sistema escribe el fichero en la carpeta seleccionada.

Prototipo de Interfaz

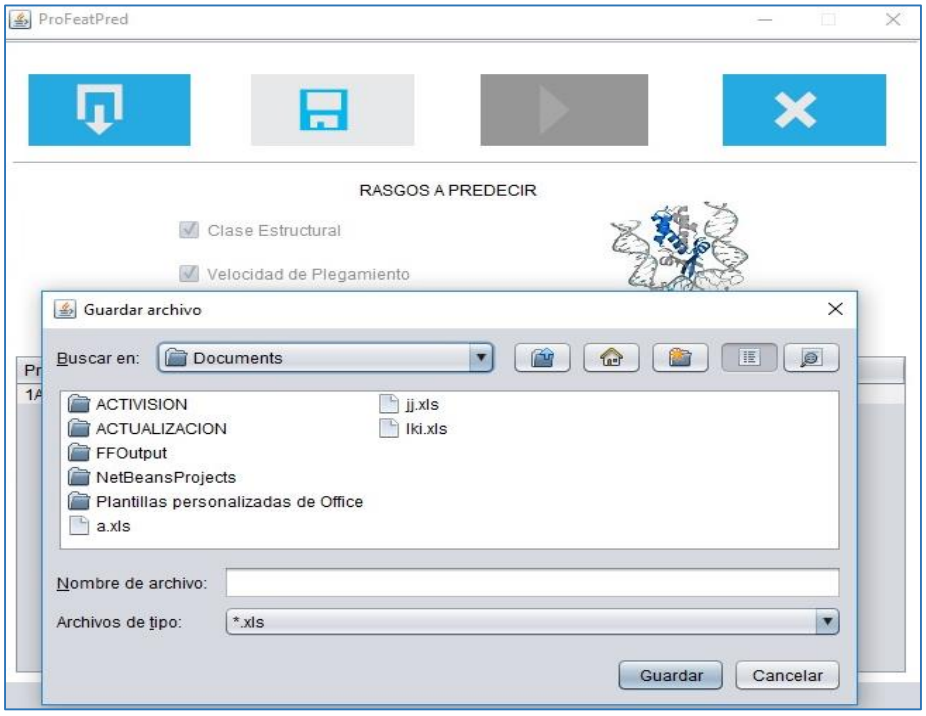


Fig. 18 Guardar resultados

2.2.3. Diseño

Técnica de Representación del Conocimiento

Después de haber recolectado, interpretado y analizado el conocimiento, se tiene la necesidad de representarlo, para realizar esta actividad, existen diversas técnicas de representación del conocimiento como son basados en marcos, basado en reglas y el método de la inducción (Durkin and Durkin 1998). Basado en marcos su utilización es apropiada si el experto describe el problema referenciando los objetos importantes y sus relaciones, además de que puede ser bien escogido si el experto considera varios objetos similares cuando resuelve el problema. Basado en reglas es conveniente si el experto discute el problema principalmente usando declaraciones tipo IF/THEN. El método de la inducción su utilización es apropiada si no existe ningún experto real en el problema, la historia de información del problema está disponible, ya que

puede usarse para derivar los procedimientos de toma de decisión automáticamente. Esta técnica es la que se utiliza en la creación del sistema experto ProFeatPred ya que existen ejemplos pasados del problema.

Técnica de Control

El encadenamiento hacia adelante es apropiado utilizarlo cuando el experto busca un camino desde el problema a la solución, va desde los hechos hasta las conclusiones que siguen a partir de los hechos.

El encadenamiento hacia atrás es una buena opción si el experto primero considera alguna conclusión o meta, luego intenta demostrarlo buscando la información de apoyo. En este caso, el experto está principalmente interesado en demostrar alguna hipótesis o recomendación (Durkin and Durkin 1998). En la investigación la técnica de control seleccionada es el encadenamiento hacia adelante debido a que primero se recolecta la información sobre el problema y luego se ve qué puede ser concluido.

Diagrama de clases del diseño

En el diagrama de clases de la Fig. 19 se muestran las capas que conforman al sistema experto, sus respectivas clases y una descripción de las mismas.

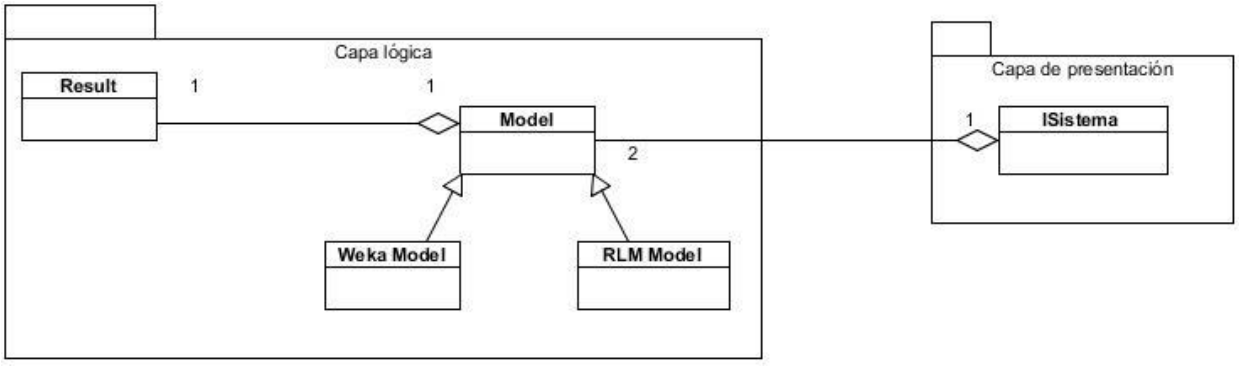


Fig. 19 Diagrama de clases del sistema experto

- ✓ *Model* es una clase abstracta en la que se obtiene una predicción.
- ✓ *Weka Model* es una clase que hereda de la clase *Model* obtenido por el software Weka y se utiliza para predecir la clase estructural.
- ✓ *RLM Model* representa un modelo de regresión lineal múltiple y su objetivo es predecir la velocidad de plegamiento en proteínas.
- ✓ *Result* es la clase que tiene como propósito almacenar los resultados de las predicciones.

- ✓ *ISistema* es la clase que permite al usuario interactuar con el sistema.

Desarrollo de Interfaces

Como resultado de la implementación se muestran la interfaz del sistema en la Fig. 20.

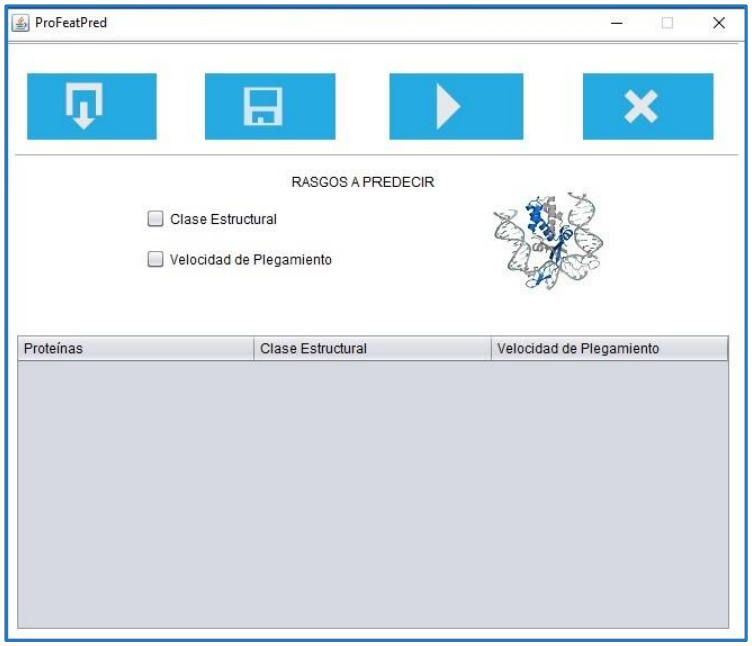


Fig. 20 Vista principal de ProFeatPred

2.2.4. Diagrama de componentes

En la Fig. 21 se muestra el diagrama de componentes del sistema experto y posteriormente se describen los componentes de dicho diagrama.

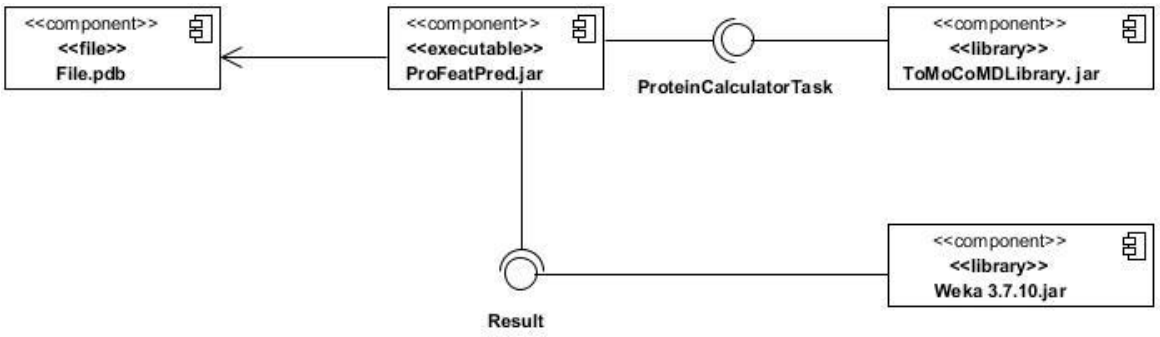


Fig. 21 Diagrama de componentes del sistema experto

- ✓ El componente *File.pdb* contiene la proteína, con todos sus átomos, que serán procesados con el componente ejecutable *ProFeatPred.*jar

- ✓ El componente *ProFeatPred.jar* es un ejecutable que permite cargar los ficheros y predecir la clase estructural y la velocidad de plegamiento, para el cálculo de los descriptores de contacto se invoca la clase *ProteinCalculatorTask* que está incluida en la biblioteca *ToMoCoMDLibrary* y mediante la clase *Result* se almacenan los resultados obtenidos según los modelos.
- ✓ El componente *ToMoCoMDLibrary.jar* es una biblioteca que permite calcular los descriptores moleculares.
- ✓ El componente *Weka 3.7.10.jar* es una biblioteca que se utiliza para obtener los modelos de clasificación.

2.2.5. Estándar de codificación

El estándar de codificación utilizado para la confección de ProFeatPred es el estándar definido en la *Java Code Conventions* que plantea:

- ✓ Las variables de clase (static) deben estar organizadas; primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso) y después las private.
- ✓ Los métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.
- ✓ Comentario de implementación de la clase o interface si fuera necesario se plantean de esta forma (/*...*/). Este comentario debe contener cualquier información aplicable a toda la clase o interface que no era apropiada para estar en los comentarios de documentación de la clase o interface.
- ✓ Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.
- ✓ Se recomienda una declaración por línea, ya que facilita los comentarios.
- ✓ Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.
- ✓ Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.

2.3. Desarrollo de los modelos de clasificación y regresión

2.3.1. Modelo de clasificación

Descripción del conjunto de datos

Para el desarrollo de los modelos de clasificación se utilizó el conjunto de datos propuesto en (Chou 1999) el cual está compuesto por 204 proteínas de las cuales: 52 son *All-α*, 61 *All-β*, 45 *α/β* y 46 *α+β*. Con el propósito de obtener los modelos de clasificación y evaluar su capacidad predictiva se utilizó una división realizada previamente; para detalles sobre el procedimiento utilizado para realizar dicha división y la composición cuantitativa y cualitativa de los conjuntos resultantes remitirse al material suplementario de (Marrero-Ponce et al. 2015). De esta forma, el conjunto original quedó dividido en series de entrenamiento (SE) y predicción (SP).

Desarrollo de los modelos de clasificación

Los modelos de clasificación se obtuvieron por medio de las técnicas: RF, K-NN y MLP, las cuales están implementadas en el software WEKA 3.7.10 utilizando descriptores de contacto calculados con el software ToMoCoMD-CAMPS. Este software genera un número considerable de DMs por la posibilidad de combinar sus diferentes parámetros (formas algebraicas, enfoques matriciales, propiedades de la cadena lateral de aminoácidos, operadores de agregación, etc.), por esta razón, se aplicó un procedimiento de selección de rasgos, el cual se describe a continuación:

1. Mezclar todos los ficheros obtenidos.
2. Seleccionar los 100 DMs de mayor variabilidad por Entropía de Shannon con el software IMMAN 1.0 (Urias, Barigye, Marrero-Ponce, García-Jacas, Valdes-Martiní and Perez-Gimenez 2015).
3. Adicionar al fichero generado en el paso 2, la variable respuesta (clase estructural).
4. Aplicar al fichero obtenido en el paso 3, la técnica de selección de atributos Correlation Feature Selection disponible en el software WEKA 3.7.10.

2.3.2. Modelo de regresión

Descripción los conjuntos de entrenamiento y prueba

Para el desarrollo y evaluación de los modelos de regresión, se emplearon conjuntos de 80 (Ouyang 2008) y 16 (Ruiz-Blanco, Marrero-Ponce, Prieto, Salgado, García and Sotomayor-Torres 2015a) proteínas como SE y SP, respectivamente. Resulta importante señalar que de la SE fue excluida la proteína con identificador "2BLM" por contener únicamente las coordenadas espaciales de sus átomos Cα.

Desarrollo de los modelos de regresión

Los modelos Regresión Lineal Múltiple (RLM) se obtuvieron con el software MobyDigs 1.0 (Todeschini et al. 2005) empleando descriptores 3D-proteicos calculados con el software ToMoCoMD-CAMPS MuLiMs-MCoMPAs. Para realizar el estudio, se empleó el conjunto de proyectos utilizado en la clasificación

estructural de proteínas, por lo que fue necesario planear y aplicar un procedimiento para reducir la alta dimensión de DMs. Se usó en el mismo procedimiento del subepígrafe anterior, además de:

1. Aplicar los siguientes filtros al fichero generado anteriormente:
    a) Eliminar DMs con **correlación** mayor o igual a 0.95.
    b) Eliminar DMs con **kurtosis** mayor que 8.
    c) Eliminar DMs con **entropía estandarizada** menor que 0.3.

Conclusiones del capítulo

Después de haber realizado el análisis y diseño del módulo descriptores de contacto del software ToMoCoMD-CAMPS y del sistema experto ProFeatPred, se arribaron a las siguientes conclusiones:

- ✓ En el análisis del módulo descriptores de contacto se identificaron 5 conceptos que permitieron realizar el modelo de dominio, se realizó el levantamiento de 11 requerimientos funcionales y 5 no funcionales, sirviendo de base para elaborar el diagrama de casos de uso del sistema y realizar la descripción del caso de uso crítico Calcular Descriptores de Contacto.

- ✓ La arquitectura de base definida fue el estilo en dos capas el cual permitió identificar los elementos implicados en el desarrollo de la solución. De igual manera, se reconocieron las clases con las que trabaja el descriptor, quedando plasmado así el diagrama de clases. Durante el diseño se utilizaron patrones heredados del software ToMoCoMD-CAMPS y en la etapa de implementación quedó elaborado el diagrama de componentes y definido el estándar de codificación.

- ✓ El análisis del sistema experto se realiza siguiendo la metodología John Durkin, cumpliendo sus fases y tareas, en la fase de Evaluación se estudia la viabilidad del proyecto, demostrando que este es viable, en la fase de Adquisición del Conocimiento se identificaron 5 conceptos que permitieron la realización del modelo de dominio, de igual forma se realizó el levantamiento de 4 requerimientos funcionales y 5 no funcionales lo que permitió elaborar el diagrama de casos de uso del sistema y realizar la descripción de los casos de usos del mismo. En la fase de Diseño se agruparon por capas las clases del sistema quedando conformado el diagrama de clases y por último quedó elaborado el diagrama de componentes y definido el estándar de codificación.

- ✓ Se definen los conjuntos de datos y los procedimientos utilizados para desarrollar los modelos de clasificación y regresión.

Capítulo 3: Evaluación del Sistema Experto para la Predicción de Propiedades Biológicas en Proteínas

En este capítulo se evalúa el desempeño de los modelos de clasificación y regresión mostrando los parámetros estadísticos asociados a estos modelos. Se valida el cálculo de los descriptores de contacto realizándole la prueba del camino básico, para ello se calcula el grafo de Complejidad Ciclomática. Además, se muestran las pruebas realizadas al sistema experto a nivel de unidad y de sistema.

3.1. Evaluación del desempeño de los modelos de clasificación y regresión

3.1.1. Evaluación del desempeño de los modelos de clasificación

La evaluación del desempeño de los modelos de clasificación obtenidos se realiza mediante los siguientes parámetros Exactitud o Precisión (da Silveira, Pires, Minardi, Ribeiro, Veloso, Lopes, Meira, Neshich, Ramos and Habesch) que es el porcentaje de casos (proteínas) clasificadas correctamente (Baldi et al. 2000), la Sensibilidad (Baldi, Brunak, Chauvin, Andersen and Nielsen 2000) que es la probabilidad de clasificar correctamente un caso positivo (Baldi, Brunak, Chauvin, Andersen and Nielsen 2000) y la Especificidad (ESP) que es la probabilidad de que una predicción positiva sea correcta (Baldi, Brunak, Chauvin, Andersen and Nielsen 2000).

En las tablas 6,7 y 8 se muestran los parámetros estadísticos asociados a los modelos obtenidos. Los mismos muestran como resultado de exactitud global 95.97%(143/149) para el MLP, 94.63% (141/149) en el K-NN y 93.96% (140/149) en el Random Forest. Con el objetivo de evaluar la robustez de los modelos, se aplicó la estrategia de validación interna 10-fold cross validation sobre la serie de entrenamiento (149 proteínas) obteniendo una exactitud global de 94.57% (141/149), 92.75% (139/149) y 91.85% (135/149) en los modelos basados en MLP, K-NN y Random Forest; respectivamente. Valores elevados de exactitud global son indicativos de la robustez en un modelo (Tropsha 2010).

Además, en cuanto a la capacidad predictiva de los modelos, se percibe que el modelo desarrollado con MLP exhiben una precisión global de 96.59% (49/55), K-NN clasifica correctamente el 92.73% (51/55) de los casos y Random Forest clasifican correctamente en un 96.36% (53/55) de los casos, quedando seleccionado como mejor modelo MLP.

Tabla 6. Parámetros estadísticos del modelo obtenido usando MLP

Conjunto	Exactitud global Q (%)	Sensibilidad (%)		Especificidad (%)	
		Clases		Clases	
Serie de entrenamiento	95.97	α+β	91.20	α+β	97.40
		α/β	100	α/β	99.20

		All-β	100	All-β	100
		All-α	92.10	All-α	98.20
Serie de Prueba	96.59	**Clases**		**Clases**	
		α+β	91.40	α+β	100
		α/β	100	α/β	100
		All-β	100	All-β	95.20
		All-α	100	All-α	90.20

Tabla 7. Parámetros estadísticos del modelo obtenido usando K-NN

Conjunto	**Exactitud global Q (%)**	**Sensibilidad (%)**		**Especificidad (%)**	
Serie de entrenamiento	94.63	**Clases**		**Clases**	
		α+β	85.30	α+β	97.40
		α/β	100	α/β	99.20
		All-β	100	All-β	98.00
		All-α	92.10	All-α	98.20
Serie de prueba	92.73	**Clases**		**Clases**	
		α+β	83.33	α+β	95.30
		α/β	100	α/β	100
		All-β	100	All-β	97.60
		All-α	85.70	All-α	97.60

Tabla 8. Parámetros estadísticos del modelo obtenido usando Random Forest

Conjunto	**Exactitud global Q (%)**	**Sensibilidad (%)**		**Especificidad (%)**	
Serie de entrenamiento	93.96	**Clases**		**Clases**	
		α+β	91.20	α+β	95.70
		α/β	96.60	α/β	100
		All-β	97.90	All-β	100
		All-α	89.50	All-α	98.20
Serie de prueba	96.36	**Clases**		**Clases**	
		α+β	91.70	α+β	100
		α/β	100	α/β	100
		All-β	92.90	All-β	90.20
		All-α	100	All-α	90.20

3.1.2. Evaluación del desempeño de los modelos de regresión

Para estimar los modelos de regresión se toman en consideración dos aspectos esenciales: el desempeño interno (bondad de ajuste y robustez) y externo (capacidad de predicción) como se sugiere en (http://www.oecd.org/chemicalsafety/risk-assessment/37849783.pdf). Las medidas que a continuación se describen quedaron clasificadas en tres grupos: (ajuste, robustez y predicción externa), en correspondencia con lo expresado anteriormente.

Medida de ajuste: Coeficiente de correlación cuadrado o coeficiente de determinación (R^2): este parámetro estima la proporción de la variable respuesta explicada por la regresión. Es decir, si no existe relación lineal entre la variable dependiente (endpoint) y las variables independientes, entonces R^2=0; por otra parte, si existe un ajuste perfecto, entonces R^2=1.

Medidas de robustez: Una de las técnicas empleadas para evaluar la robustez es la denominada validación cruzada (VC) por re-muestreo (bootstrapping) (Wehrens et al. 2000), en la cual se determinan aleatoriamente conjuntos de entrenamiento y de prueba. La premisa básica del re-muestreo es que los datos deben ser representativos de la población de que fueron extraídos. En este procedimiento el conjunto de entrenamiento preserva el tamaño *(n)* original del conjunto de datos y se conforma mediante la selección de *n* objetos con repetición, por su parte el conjunto de prueba está constituido por objetos no seleccionados para formar parte del entrenamiento. Este procedimiento se repite varias veces y en cada iteración se calcula el PRESS (ver Ec. 12), para finalmente determinar el poder predictivo promedio (Q^2_{boot}). Otra técnica ampliamente utilizada es la prueba de permutación de las respuestas o *Y-scrambling* para identificar modelos basados en correlación aleatoria, es decir, modelos cuyas variables independientes están correlacionadas aleatoriamente con la(s) variable(s) respuesta(s). La prueba se realiza determinado la calidad del modelo (comúnmente o Q^2) modificando aleatoriamente la secuencia del vector respuesta, es decir, asignando a cada compuesto una respuesta seleccionada aleatoriamente del verdadero conjunto de respuestas. Si el modelo original no presenta correlación aleatoria, entonces existe una diferencia notable entre la "calidad" del modelo original en comparación con el modelo obtenido con las variables respuestas permutadas.

Medidas de predicción externa: La validación cruzada dejando uno fuera *leave-one-out* (LOO) es el procedimiento de VC más sencillo. En este caso, dados *n* compuestos (proteínas), se calculan *n* modelos reducidos, es decir, obtenidos con n-1 casos. Estos modelos se usan para predecir la respuesta del compuesto excluido. El poder predictivo se calcula como la suma al cuadrado de las diferencias entre la variable experimental y la respuesta estimada. Matemáticamente se define mediante la siguiente fórmula:

$$Q^2_{loo} = 1 - \frac{PRESS}{TSS} = 1 - \frac{\sum_{i=1}^{nce}\left(Y_i - Y_i^{''}\right)^2}{\sum_{i=1}^{nce}\left(Y_i - \tilde{Y}\right)^2} \quad \textbf{(12)}$$

donde, PRESS es la suma de los cuadrados de los residuos de la predicción, *TSS* es la suma total de cuadrados, *nce* es el número de elementos del conjunto de entrenamiento, *Yi, Y''* es la respuesta observada y estimada del *i-ésimo* elemento respectivamente $\tilde{Y}$ es el promedio de las respuestas del conjunto de entrenamiento. El parámetro Q^2_{loo} varía en el intervalo [0-1]. Los modelos con valores de Q^2_{loo} próximos a 0 presentan un bajo poder predictivo, mientras que los modelos con valores cercanos a 1 tienen una alta capacidad de predicción.

La validación externa permite evaluar si los modelos obtenidos son generalizables a nuevos compuestos químicos y de esta forma evaluar el verdadero poder predictivo de los mismos. Para esto se divide la base en dos conjuntos: la serie de entrenamiento para construir el modelo y la serie de predicción para evaluar el modelo obtenido. El estadístico utilizado con este fin se denomina Q^2_{ext} y se calcula mediante la expresión:

$$Q^2_{ext} = 1 - \frac{\sum_{i=1}^{nce}\left(Y_i - Y_i^{''}\right)^2}{\sum_{i=1}^{nce}\left(Y_i - \tilde{Y}\right)} \quad \textbf{(13)}$$

En general, se obtienen modelos de dos a siete variables como se evidencia en el Anexo 2, observándose que los valores obtenidos por la técnica "bootstrapping" (Q^2_{boot}) superan el 56% de la varianza total, por lo tanto, los modelos desarrollados pueden considerarse como robustos o con buena capacidad de predicción interna. Así mismo los coeficientes determinados en el procedimiento "Y-scrambling" [a(Q^2)] presentan en todos los casos valores inferiores a 0.180 indicando que la correlación entre las variables independientes y la variable modelada presenta un ínfimo grado de aleatoriedad. Por otra parte, como el número de proteínas del conjunto de entrenamiento es pequeño (inferior a 100), el parámetro Q^2_{loo} puede ser considerado como un indicador de la capacidad predictiva externa de los modelos (Todeschini and Consonni 2009; Tropsha et al. 2003). En este sentido, se observa que los modelos obtenidos explican como mínimo el 55% de la varianza total. En cuanto al coeficiente de correlación cuadrado (R^2) el cual estima la proporción de la variable respuesta explicada por la regresión se observan valores que oscilan los 0.595 y 0.825 existiendo una aceptable relación lineal entre la variable dependiente y las variables independientes. Adicionalmente, si se examina el desempeño en el conjunto externo se puede decir que los modelos obtenidos poseen un

buen poder predictivo, puesto que los valores del estadístico Q^2_{ext} se encuentran en el orden de los 0.596, lo que indica que los modelos obtenidos explican aproximadamente el 60% de la varianza total. Se selecciona como mejor modelo de regresión el de tres variables del cual se presentan los parámetros estadísticos en la Tabla 9.

Tabla 9. Parámetros estadísticos del modelo de Regresión Lineal Múltiple

Dim	R^2	Q^2_{loo}	Q^2_{boot}	$a(Q^2)$	Q^2_{ext}	$SDEP_{ext}$	Modelo
3	0.7004	0.6697	0.6657	-0.158	0.5926	1.907	**ln_kf** = 0.02316(±0.08675)CA_Q2_F_M34_SS1 _T_KA_EPS_MCoMPAs + 0.00455(±0.02263)CA_I50_B_M35_NS1 _T_KA_Z2-GCP1_MCoMPAs + 0.18834(±0.79698)CA_I50_B_M35_NS1 _T_KA_Z3-PTT_MCoMPAs + 55.2354(±9.72145)

3.2. Pruebas

La meta de probar es encontrar errores, y una buena prueba es aquella que tiene una alta probabilidad de encontrar uno. Por tanto, un sistema basado en computadora o un producto debe diseñarse e implementarse teniendo en mente la comprobabilidad. Al mismo tiempo, las pruebas en sí mismas deben mostrar un conjunto de características que logren la meta de encontrar la mayor cantidad de errores con el mínimo esfuerzo (Pressman 2010).

3.2.1. Pruebas del módulo descriptores de contacto

Validación de los descriptores de contacto

En la Fig. 22, se muestran las coordenadas atómicas de un pequeño péptido de cinco aminoácidos y se fija como intervalo de distancia 4-5.9 Å. Posteriormente, se calcula la distancia euclidiana interatómica, si la distancia entre los átomos i y j cae en el intervalo definido, entonces la entrada ij de la matriz toma valor 1 y 0 en otro caso, como se muestra en la Fig. 23.

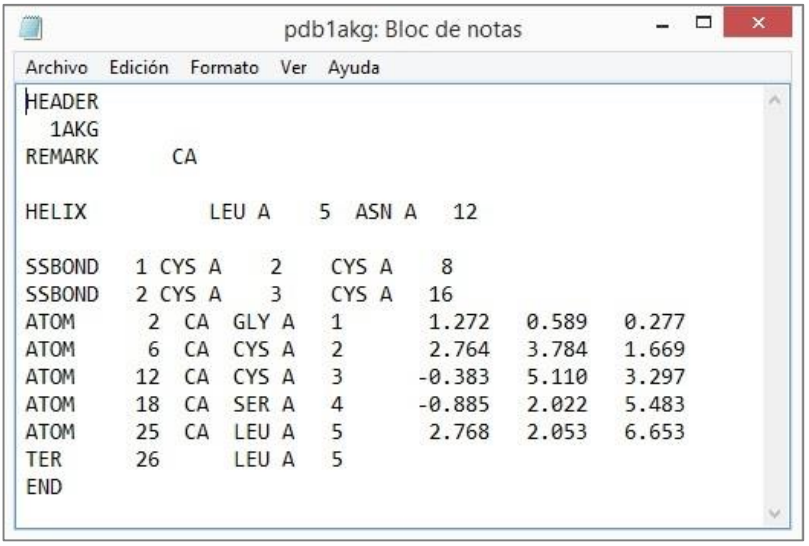


Fig. 22 Péptido de ejemplo

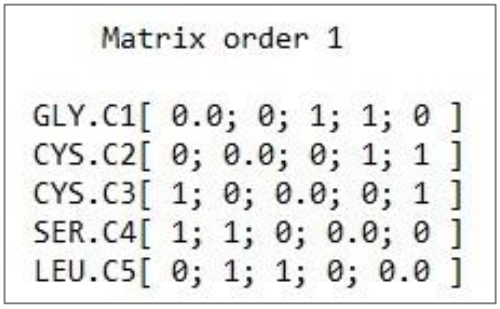


Fig. 23 Matriz de contacto no estocástica

Prueba de ruta básica al método calculateDescriptor de la clase AlgebraicDescriptorsDuples del módulo descriptores de contacto

La prueba de ruta o trayectoria básica es una técnica de prueba de caja blanca, esta prueba permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman 2010).

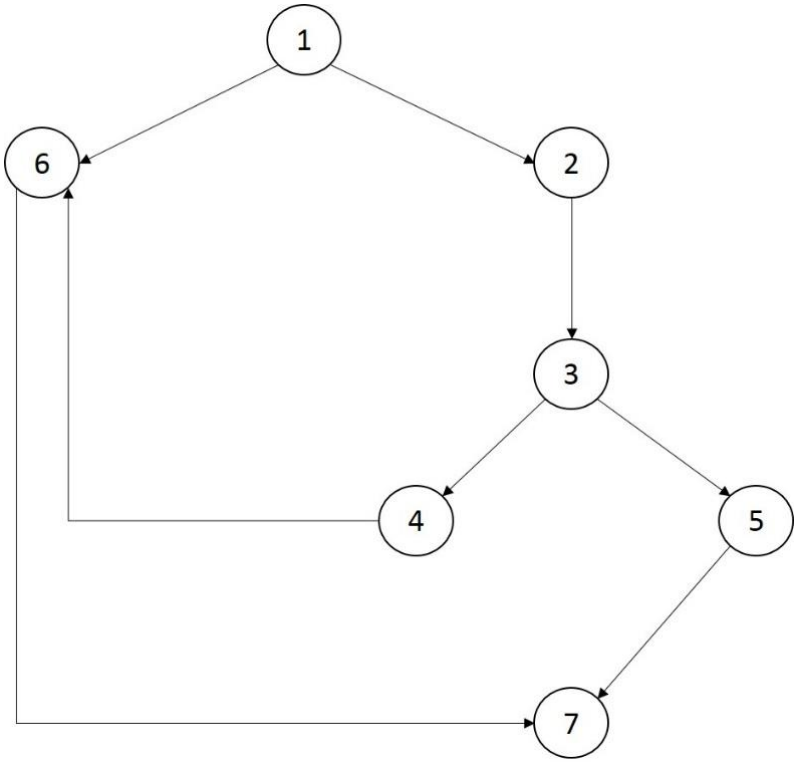


Fig. 24 Grafo del Flujo del método calculateDescriptor

A continuación, se hace el cálculo de la complejidad ciclomática, que no es más que es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y le brinda una cota superior para el número de pruebas que debe realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez.

V (G) = cantidad de aristas - cantidad de nodos + 2
V (G) = 8 - 7 + 2 = 3.
V (G) = cantidad de regiones = 3.

Caminos de prueba

Camino 1: 1-2-3-4-6-7
Camino 2: 1-2-3-5-7
Camino 3: 1-6-7

En la tabla 10 se muestra la descripción de cada camino.

Tabla 10. Descripción de los caminos

Camino	Descripción	Entrada	Salida
1	El cliente ha configurado previamente un descriptor de contacto y se dispone a calcularlo	Selecciona la métrica de contacto o ninguna	Descriptores de contacto o en otro caso excepción
2	El cliente ha configurado previamente un descriptor de contacto y se dispone a calcularlo	Selecciona la métrica de contacto y el orden no está definido en 1	Descriptores de contacto o en otro caso excepción
3	El cliente ha configurado previamente un descriptor de contacto y se dispone a calcularlo	Configura manualmente la distancia inter-atómica o lo deja en blanco	Descriptores de contacto o en otro caso excepción

Para comprobar el correcto funcionamiento del método calculateDescriptor y validar los caminos, se realizan pruebas utilizando el entorno de desarrollo NetBeans. Se crearon casos de prueba JUnit (del inglés, *JUnit Test Case*), comprobando de esta forma que los procedimientos que corresponden al CU crítico funcionan correctamente, como se exhibe en la Fig. 25.

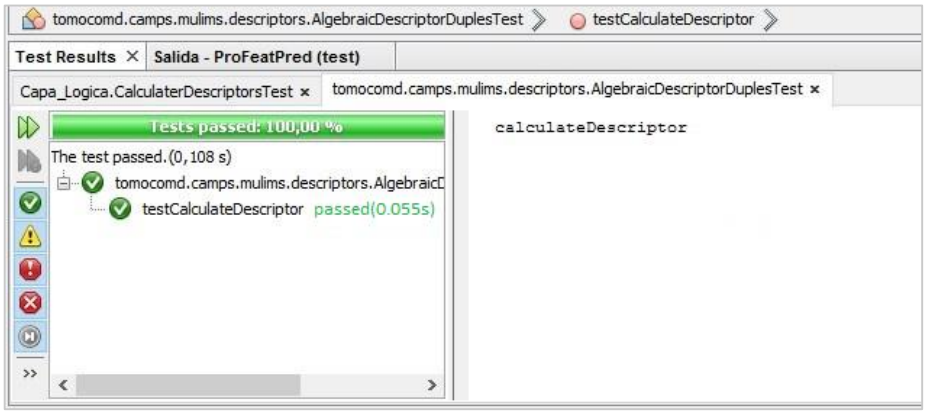


Fig. 25 Resultado de la prueba en el JUnit para el método calculateDescriptor

3.2.2. Validación del sistema

En este epígrafe se realizarán las validaciones al sistema y las pruebas pertinentes que comprueben la calidad del software, de esta manera se comprueba si los resultados obtenidos a partir de la entrada de los datos satisfacen los resultados esperados. ProFeatPred se probó en los niveles de unidad y sistema.

Las pruebas a nivel de unidad se centran en el proceso de verificación en la menor unidad del diseño del software (el componente de software o módulo) (Sommerville 2005), de esta manera se comprueban todos los flujos de control. Se realiza la prueba de caja blanca, utilizando el entorno de desarrollo NetBeans se

crean casos de prueba JUnit, comprobando de esta forma que los procedimientos que corresponden a los CU críticos “Predecir velocidad de plegamiento” y “Predecir clase estructural” funcionan correctamente, como se muestra en la Fig.26.

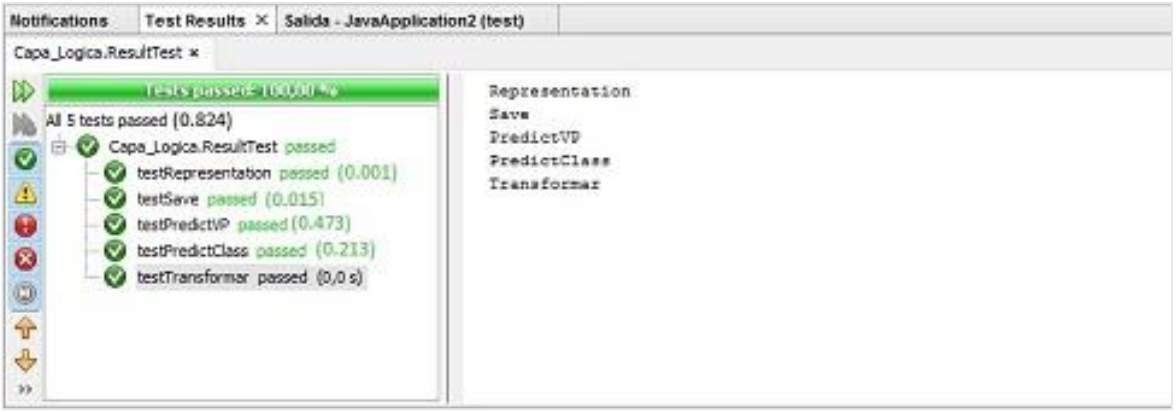


Fig. 26 Resultado de la prueba en el JUnit en los métodos PredictVP y PredictClass

Las pruebas a nivel de sistema se refieren al comportamiento del sistema integrado. Estas se aplican generalmente para probar los requerimientos de la solución (Sommerville 2005). Para este nivel se efectúa la prueba de funcionalidad de caja negra y la técnica de partición de equivalencia, esta última divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Al aplicar la técnica de caja negra, se derivan un conjunto de casos de prueba, que consisten en la creación de un grupo de variables o condiciones mediante las cuales se podrá determinar si los requisitos de una aplicación son parciales o completamente satisfactorios y para identificar los posibles fallos de implementación. En la Tabla 11 se muestra la descripción del caso de prueba “Predecir clase estructural”.

Tabla 11. Descripción del caso de prueba “Predecir clase estructural”

Escenario	Clase estructural	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Predecir clase estructural	V	En este escenario el actor selecciona la opción clase estructural para realizar la predicción de este rasgo en la(s) proteína(s).	El sistema muestra en la tabla la clase estructural de la proteína	Se selecciona la opción clase estructural. Se selecciona la opción predecir. Se muestra el resultado en la tabla.
EC 1.2 No predice la clase estructural	I	En este escenario el actor no selecciona la	El sistema muestra un mensaje de	Se selecciona la opción predecir.

		opción clase estructural e intenta predecir este rasgo en la(s) proteína(s).	notificación informando al usuario que no ha seleccionado el rasgo a predecir	

Las pruebas del sistema ejecutadas mediante los casos de prueba, mostraron siete No Conformidades (NC) en la primera iteración como se muestra en la Fig. 27. Después de corregidas las NC detectadas, se ejecutó una segunda iteración donde se comprobó la corrección de dichas NC y se detectaron otras tres Fig. 28, que posterior a su erradicación, se verificó en una tercera iteración que el sistema quedó libre de NC.
En la tabla 12, se evidencian ejemplos de las NC adquiridas y en el gráfico de barras de la Fig. 29 se muestra los resultados obtenidos y su distribución por iteraciones.

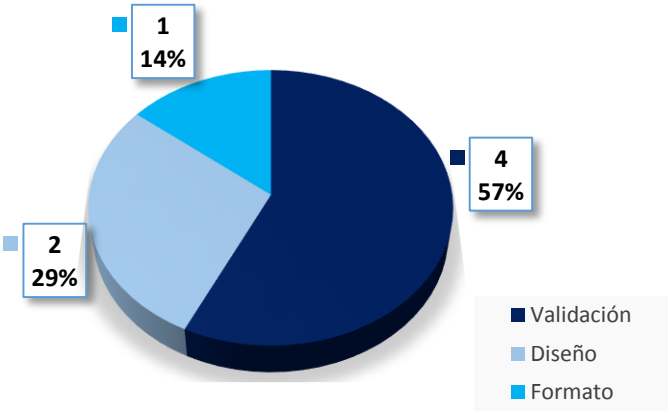


Fig. 27 No conformidades detectadas en la primera iteración

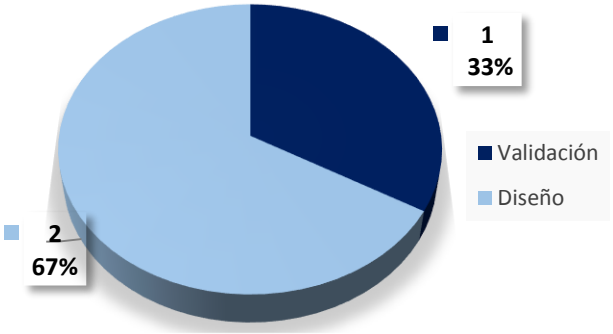


Fig. 28 No conformidades detectadas en la segunda iteración

Tabla 12. Ejemplo de No conformidades detectadas

Elemento	No	No Conformidad	Ubicación	Etapa de detección	Clasificación
Aplicación	1	El campo nombre de archivo admite cualquier tipo de archivo	ProFeatPred/Cargar archivo	Pruebas de funcionalidad	Significativas
Aplicación	2	No se muestra ninguna notificación de error al usuario intentar predecir algún rasgo sin antes haberlo seleccionarlo.	ProFeatPred/Predecir Propiedades	Pruebas de funcionalidad	Significativas
Aplicación	3	Error ortográfico (Ausencia de tildes) en la palabra predicción.	ProFeatPred/Predecir Propiedades	Pruebas de funcionalidad	Significativas
Aplicación	4	Error ortográfico (Ausencia de tildes) en la palabra proteínas.	ProFeatPred	Pruebas de funcionalidad	Significativas

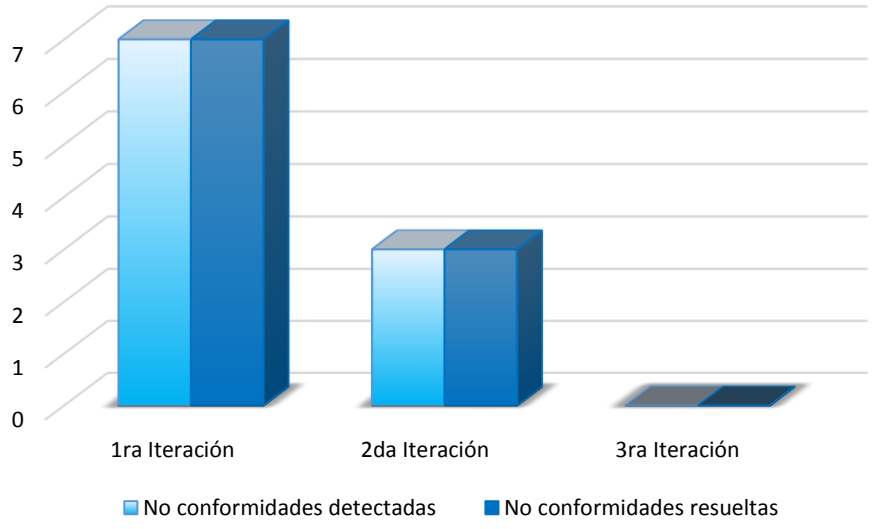


Fig. 29 No conformidades

Conclusiones del capítulo

Después de haber realizado las pruebas del módulo descriptores de contacto del software ToMoCoMD-CAMPS y las pruebas del sistema experto ProFeatPred, se arribaron a las siguientes conclusiones:

- ✓ En la evaluación de los modelos de clasificación y regresión, se examinó el desempeño de estos, quedando seleccionado el modelo obtenido con la técnica Perceptrón Multicapa para la discriminación estructural y como modelo de regresión lineal de tres variables.

- ✓ Durante la prueba de validación del descriptor, se evidencia que este se calcula correctamente las distancias euclidianas, dependiendo del intervalo definido por el investigador, de esta forma queda demostrado cómo se obtiene la matriz de contacto.

- ✓ Se realizó la prueba de la ruta básica, calculándose la complejidad ciclomática, alcanzando de esta forma resultados satisfactorios.

- ✓ Al sistema experto se le realizaron pruebas en los niveles de unidad y de sistema. Las pruebas de unidad al que el sistema fue sometido devolvieron buenos resultados, a nivel de sistema se realizaron pruebas de caja negra, aplicando técnica de validación cruzada, se detectaron siete no conformidades y fueron corregidas.

Conclusiones generales

Al concluir al presente trabajo se arriban a los siguientes planteamientos:

- ✓ Se desarrolla un módulo para el cálculo de descriptores de contacto basado en las formas algebraicas bilineales.
- ✓ Se desarrolla un sistema experto basado en descriptores de contacto siguiendo la Metodología de John Durkin, el cual permite predecir la clase estructural y la velocidad de plegamiento en proteínas, proporcionando que el usuario interactúe de forma amigable y sencilla.
- ✓ Se desarrollaron y evaluaron modelos de clasificación y regresión basados en las técnicas Perceptrón Multicapa y Regresión Lineal Múltiple.
- ✓ Se validaron las funcionalidades implementadas por medio de pruebas a nivel de unidad y sistema de forma automatizada.
- ✓ Como hipótesis de investigación se planteó que si se utiliza un sistema experto basado en modelos desarrollado con descriptores de contacto bilineales, entonces es posible predecir velocidad de plegamiento y clase estructural de proteínas. Los resultados alcanzados evidencian que se acepta la hipótesis propuesta.

Recomendaciones

Al concluir el trabajo quedan aspectos que deben ser tratados posteriormente en ese sentido se recomienda:

- ✓ Considerar otras representaciones de contacto usando distancia topológica, es decir separación entre aminoácidos en secuencia y combinar con contacto geométrico que fue el que se realizó en este trabajo.
- ✓ Desarrollar un web server para poner el sistema experto en la web.
- ✓ Utilizar los descriptores propuestos para modelar otras propiedades de interés.

Referencias bibliográficas

1. ACEBEDO, J. La actualización del Modelo Económico Cubano: un reto a las empresas e instituciones. CUJAE, 2012.
2. AKKERMANS, H., G. SCHREIBER, A. ANJEWIERDEN, R. DEHOOG, et al. Knowledge engineering and management: the commonKADS methodology. In.: Cambridge: MIT Press, 1999.
3. ALVAREZ, R. L. *De la información a la informática*. Edtion ed.: Politécnico Colombiano Jaime Isaza Cadavid. Subsistema de información para la Modalidad Tecnológica, 1991.
4. BADARÓ, S., L. J. IBAÑEZ AND M. AGÜERO Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones. Ciencia y Tecnología, 2013, 1(13).
5. BALDI, P., S. BRUNAK, Y. CHAUVIN, C. A. ANDERSEN, et al. Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics, 2000, 16(5), 412-424.
6. BECERRA, R. A. Sistemas Expertos para la realización de diagnóstico de trastornos neuromusculares con electromiografía Mendoza, 14. Universidad del Aconcagua, 2011.
7. BOOCH, G., J. RUMBAUGH, I. JACOBSON, J. S. E. MARTÃ-NEZ, et al. *El lenguaje unificado de modelado*. Edtion ed.: Addison-Wesley, 1999.
8. BORROTO, Ó. M. R., Y. H. DÍAZ, J. M. G. DE LA VEGA, R. GRAU, et al. Perspectiva general sobre el proceso de desarrollo de fármacos y las técnicas de cribado virtual basadas en la similitud molecular. In *Anales de la Real Academia Nacional de Farmacia.* 2013, vol. 79.
9. BOYLE, J. Lehninger principles of biochemistry: Nelson, D., and Cox, M. In.: Wiley Online Library, 2005.
10. BREIMAN, L. Random Forests. Machine Learning, 45(1), 5-32.
11. BRINDA, K. AND S. VISHVESHWARA A network representation of protein structures: implications for protein stability. Biophysical journal, 2005, 89(6), 4159-4170.
12. BUCHANAN, B. G., D. BARSTOW, R. BECHTAL, J. BENNETT, et al. Constructing an expert system. Building expert systems, 1983, 50, 127-167.
13. CARLOS SOTO, M. Sistema Experto de Diagnóstico médico del síndrome de Guillian Barre. Portal del Sistema de Bibliotecas de la UNMSM, 2005.
14. CONTRERAS-TORRES, E., Y. MARRERO-PONCE AND C. R. GARCÍA-JACAS Nueva codificación de la estructura tridimensional de proteínas.
15. CUMBAL, T., W. CELIANO AND E. J. TORRES CAÑIZARES Elaboración de un manual de la plataforma Netbeans Ide para la Disicom 2009.
16. CHAPMAN, P., J. CLINTON, R. KERBER, T. KHABAZA, et al. CRISP-DM 1.0 Step-by-step data mining guide 2000.

17. CHOU, K.-C. A key driving force in determination of protein structural classes. Biochemical and biophysical research communications, 1999, 264(1), 216-224.
18. CHOU, K.-C. Some remarks on protein attribute prediction and pseudo amino acid composition. Journal of theoretical biology, 2011, 273(1), 236-247.
19. CHOU, K.-C. Impacts of bioinformatics to medicinal chemistry. Medicinal Chemistry, 2015, 11(3), 218-234.
20. CHOU, K.-C. AND H.-B. SHEN FoldRate: A web-server for predicting protein folding rates from primary sequence. The Open Bioinformatics Journal, 2009, 3(2), 31-50.
21. DA SILVEIRA, C. H., D. E. PIRES, R. C. MINARDI, C. RIBEIRO, et al. Protein cutoff scanning: A comparative analysis of cutoff dependent and cutoff free methods for prospecting contacts in proteins. Proteins: Structure, Function, and Bioinformatics, 2009, 74(3), 727-743.
22. DEITEL, P. J. *Como programar en Java*. Edtion ed.: Pearson Educación, 2004. ISBN 9702605180.
23. DI PAOLA, L., M. DE RUVO, P. PACI, D. SANTONI, et al. Protein contact networks: an emerging paradigm in chemistry. Chemical Reviews, 2012, 113(3), 1598-1613.
24. DURKIN, J. AND J. DURKIN *Expert systems: design and development*. Edtion ed.: Prentice Hall PTR, 1998. ISBN 0023309709.
25. ESTRADA, E. Characterization of the folding degree of proteins. Bioinformatics, 2002, 18(5), 697-704.
26. FARISELLI, P., P. L. MARTELLI, C. SAVOJARDO AND R. CASADIO INPS: predicting the impact of non-synonymous variations on protein stability from sequence. Bioinformatics, 2015, 31(17), 2816-2821.
27. GIBAS, C. AND P. JAMBECK *Developing bioinformatics computer skills*. Edtion ed.: " O'Reilly Media, Inc.", 2001. ISBN 1565926641.
28. GLEZ, O. R. A., Y. B. R. BLANCO, G. A. CHAPIN AND Z. P. RODRIGUEZ 2016. ProtDCal aplicación 3: Desarrollo de modelos de identificación de enzimas mediante nuevos descriptores 3D de proteínas ProtDCal application 3: Development of models for enzymes identification based on novel protein structure descriptors. In *Proceedings of* 2016.
29. GONZÁLEZ-DÍAZ, H., S. VILAR, L. SANTANA AND E. URIARTE Medicinal chemistry and bioinformatics-current trends in drugs discovery with networks topological indices. Curr. Top. Med. Chem., 2007, 7(10), 1015-1029.
30. GONZÁLEZ-DÍAZ, H., Y. GONZÁLEZ-DÍAZ, L. SANTANA, F. M. UBEIRA, et al. Proteomics, networks and connectivity indices. Proteomics, 2008, 8(4), 750-778.

31. GRISONI, F., V. CONSONNI, M. VIGHI, S. VILLA, et al. Expert QSAR system for predicting the bioconcentration factor under the REACH regulation. Environmental research, 2016, 148, 507-512.
32. GROVER, M. D. A Pragmatic Knowledge Acquisition Methodology. In *IJCAI.* Citeseer, 1983, vol. 83, p. 436-438.
33. HABANA, U. D. L. Bases de Datos y Sistemas Expertos 1991, Tomo II, pág. 500-507.
34. HAIR, J., R. ANDERSON, R. TATHAM AND W. BLACK. Análisis multivariante. 5ª edición. editorial Prentice Hall. In.: Madrid, 1999.
35. HAYKIN, S. AND S. SIMON Neural Networks: A Comprehensive Foundation, Prentice-Hall, 1998.
36. HERNÁNDEZ, L. R. B., D. M. PEÑA, O. R. VALDÉS AND O. M. CORNELIO Extensión de la herramienta Visual Paradigm for UML para la evaluación y corrección de Diagramas de Casos de Uso Plugin of Visual Paradigm for UML tool for evaluation and correction of Use Case Diagram 2016a.
37. HERNÁNDEZ, L. R. B., D. M. PEÑA, O. R. VALDÉS AND B. H. GONZÁLEZ Sistema para el control del cumplimiento del Proyecto Educativo en la Enseñanza Superior Cubana. Revista Científica, 2016b, 1(24).
38. HUMBERTO GONZÁLEZ-DÍAZ1, YENNY GONZÁLEZ-DÍAZ1, LOURDES SANTANA1, AND F. M. U. A. E. URIARTE1 Proteomics, networks and connectivity indices. Bioinformatics, 2008.
39. HUMBERTO GONZÁLEZ-DÍAZ*, S. V., LOURDES SANTANA AND EUGENIO URIARTE Medicinal Chemistry and Bioinformatics – Current Trends in Drugs Discovery with
40. Networks Topological Indices. Current Topics in Medicinal Chemistry, 2007.
41. JACOBSON, I., G. RUMBAUGH, J. JACOBSON, G. BOOCH, et al. *El proceso unificado de desarrollo de software/The unified software development process*. Edtion ed.: Pearson Educación, 2000. ISBN 8478290362.
42. KENDALL, J. E. *Análisis y diseño de sistemas*. Edtion ed.: Pearson educación, 2005. ISBN 9702605776.
43. L. DI PAOLA, M. D. R., P. PACI, D. SANTONI,§ AND A. GIULIANI, Protein Contact Networks: An Emerging Paradigm in Chemistry. Chemical Reviews, 2012.
44. MARRERO-PONCE, Y., E. CONTRERAS-TORRES, C. R. GARCÍA-JACAS, S. J. BARIGYE, et al. Novel 3D bio-macromolecular bilinear descriptors for protein science: Predicting protein structural classes. Journal of theoretical biology, 2015, 374, 125-137.
45. MISHRA, A., P. S. RANA, A. MITTAL AND B. JAYARAM D2N: Distance to the native. Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics, 2014, 1844(10), 1798-1807.

46. MORATE, D. G. Manual DE Weka. Disponível através do e-mail diego. garcia. morate@ mail. com, 2008.
47. OUYANG, Z. A. J. L. Predicting protein folding rates from geometric contact and amino acid sequence. Protein Science, 2008, 17(17).
48. PAOLA, L. D., G. MEI, A. D. VENERE AND A. GIULIANI Exploring the stability of dimers through protein structure topology. Current Protein and Peptide Science, 2016, 17(1), 30-36.
49. PEÑA AYALA, A. Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo. Instituto Politécnico Nacional. México, 2006.
50. PRESSMAN, R. S. Ingeniería de software. Un enfoque práctico 2010, 7ma Edición.
51. RANDIĆ, M., K. MEHULIĆ, D. VUKIČEVIĆ, T. PISANSKI, et al. Graphical representation of proteins as four-color maps and their numerical characterization. J. Mol. Graphics Modell., 2009, 27(5), 637-641.
52. ROSSINI, P. Using expert systems and artificial intelligence for real estate forecasting. In *Sixth Annual Pacific-Rim Real Estate Society Conference, Sydney, Australia.* Citeseer, 2000, p. 24-27.
53. RUIZ-BLANCO, Y. B., Y. MARRERO-PONCE, P. J. PRIETO, J. SALGADO, et al. A Hooke′ s law-based approach to protein folding rate. Journal of theoretical biology, 2015a, 364, 407-417.
54. RUIZ-BLANCO, Y. B., W. PAZ, J. GREEN AND Y. MARRERO-PONCE ProtDCal: A program to compute general-purpose-numerical descriptors for sequences and 3D-structures of proteins. BMC bioinformatics, 2015b, 16(1), 162.
55. SEDEÑO, N. S. Módulo de comportamiento emocional de los agentes autónomos para la BIAV. UCI, 2013.
56. SOMMERVILLE, I. Requerimientos del software. Ingeniería del Software, Pearson, Madrid, 2005, 110-111.
57. STEINBECK, C., Y. HAN, S. KUHN, O. HORLACHER, et al. The Chemistry Development Kit (CDK): An open-source Java library for chemo-and bioinformatics. Journal of chemical information and computer sciences, 2003, 43(2), 493-500.
58. TAHUITON MORA, J. Arquitectura de software para aplicaciones Web. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. México, DF, 2011.
59. TODESCHINI, R. AND V. CONSONNI *Handbook of Molecular Descriptors*. edited by R. MANNHOLD, H. KUBINYI AND H. TIMMERMAN. Edtion ed. D-69469 Weinheim, Federal Republic of Germany: WILEY-VCH Verlag GmbH, 2000. 667 p. ISBN 3-52-29913-0.

60. TODESCHINI, R. AND V. CONSONNI *Molecular Descriptors for Chemoinformatics*. edited by R. MANNHOLD, H. KUBINYI AND G. FOLKERS. Edtion ed. Weinheim: WILEY-VCH, 2009. 667 p. ISBN 3-52-29913-0.
61. TODESCHINI, R., V. CONSONNI, A. MAURI AND M. PAVAN. MOBYDIGS version 1.0. [Milano, 2005.
62. TODESCHINI, R., V. CONSONNI, A. MAURI, M. PAVAN, et al. MobyDigs: software for regression and classification models by genetic algorithms. Nature-inspired Methods in Chemometrics: Genetic Algorithms and Artificial Neural Networks.(Leardi R., Ed.), 2003, 141-167.
63. TORRES, E. C. Procedimiento de extracción de rasgos 3 D-proteicos basado en Álgebra Lineal: Aplicaciones en estudios bioinformáticos. Universidad Central "Marta Abreu" de Las Villas, 2016.
64. TORRES, P. L. Desarrollo de Software Orientado a Objeto usando UML. Universidad Politecnica de Valencia (UPV)–España, 2004.
65. TROPSHA, A. Best Practices for QSAR Model Development, Validation, and Exploitation. Molecular Informatics, 2010.
66. TROPSHA, A., P. GRAMATICA AND V. K. GOMBAR The importance of being earnest: Validation is the absolute essential for successful application and interpretation of QSPR models. QSAR Comb. Sci., 2003, 22, 69–77.
67. TYPE-SAFE, P. Java Code Conventions 2014.
68. URIAS, R. W. P., S. J. BARIGYE, Y. MARRERO-PONCE, C. R. GARCÍA-JACAS, et al. IMMAN: free software for information theory-based chemometric analysis. Molecular diversity, 2015, 19(2), 305-319.
69. VLISSIDES, J., R. HELM, R. JOHNSON AND E. GAMMA Design patterns: Elements of reusable object-oriented software. Reading: Addison-Wesley, 1995, 49(120), 11.
70. WEHRENS, R., H. PUTTER AND L. M. BUYDENS The bootstrap: a tutorial. Chemometrics and intelligent laboratory systems, 2000, 54(1), 35-52.
71. WITTEN, I. H. AND E. FRANK *Data Mining: Practical machine learning tools and techniques*. Edtion ed.: Morgan Kaufmann, 2005. ISBN 008047702X.
72. YAN, W., J. ZHOU, M. SUN, J. CHEN, et al. The construction of an amino acid network for understanding protein structure and function. Amino acids, 2014, 46(6), 1419-1439.
73. ZHOU, J., W. YAN, G. HU AND B. SHEN Amino acid network for the discrimination of native protein structures from decoys. Current Protein and Peptide Science, 2014, 15(6), 522-528.

Glosario de términos

Aminoácidos: Los aminoácidos son las unidades de compuestos orgánicos que, al unirse entre sí, forman las proteínas.

Bioensayos: Proceso de determinar la potencia de una sustancia o de un material a partir de las respuestas producidas en organismos biológicos.

Cadena lateral: Es la parte que diferencia a los diferentes aminoácidos entre sí.

Diana biológica: Molécula específica del organismo sobre la que un medicamento está diseñado para actuar.

Enlace peptídico: Forma largas cadenas de decenas o cientos de aminoácidos (proteínas). Se produce entre un átomo de carbono y nitrógeno, este enlace de tipo covalente, es muy estable y resistente, lo que hace posible que se puedan formar grandes moléculas.

Plegamiento: Es el proceso expedito, termodinámicamente no espontáneo (reversible) ya que durante este proceso interfieren diferentes enzimas, azúcares. Si una proteína no se pliega correctamente, la misma no será funcional y, por lo tanto, no será capaz de cumplir con su función biológica.

Proteómica. La proteómica es el estudio de la actividad de las proteínas en una célula, un tejido o un organismo determinados.

Tamizaje: Proceso para identificar compuestos líderes.

Anexo 1

Cálculo de la Viabilidad.

ASUNTOS DE VIABILIDAD DEL PROBLEMA			
Puntaje =	**Peso ***	**Valor**	**Asunto**
63	7	9	Conocimiento experto necesitado
72	9	8	Los pasos de solución de problema son definibles
42	7	6	Conocimiento simbólico usado
56	8	7	Heurísticas usadas
90	10	9	El problema es solucionable
56	8	7	Existen sistemas exitosos
81	9	9	El problema es bien enfocado
54	6	9	El problema es razonablemente complejo
42	7	6	El problema es estable
63	9	7	Conocimiento incompleto o incierto utilizado
30	5	6	No determinístico
48	6	8	Solución más de una recomendación
Puntaje Total: 697	**Peso Total: 91**		**VIABILIDAD DEL PROBLEMA = PUNTAJE TOTAL / PESO TOTAL**
			7.659
ASUNTOS DE VIABILIDAD DEL PERSONAL			
Puntaje =	**Peso ***	**Valor**	**Asunto**
			Experto del dominio
72	8	9	El experto puede comunicar el conocimiento
72	8	9	El experto puede dedicar tiempo
49	7	7	El experto es cooperativo
Puntaje Total: 193	**Peso Total: 23**		**VIABILIDAD DEL PROBLEMA = PUNTAJE TOTAL / PESO TOTAL**
			8.391
Puntaje =	**Peso ***	**Valor**	**Asunto**
			Ingeniero del conocimiento
72	8	9	Buenas habilidades de comunicación
72	9	8	Puede relacionar el problema al software
72	8	9	Tiene destrezas de programación de sistema experto
72	9	8	Puede dedicar tiempo
Puntaje Total: 288	**Peso Total: 34**		**VIABILIDAD DEL PROBLEMA = PUNTAJE TOTAL / PESO TOTAL**
			8.471

Puntaje =	Peso *	Valor	Asunto
			Usuario Final
72	9	8	El usuario final puede dedicar tiempo
56	8	7	El usuario final es receptivo al cambio
56	7	8	El usuario final es cooperativo
Puntaje Total: 184	**Peso Total: 24**		**VIABILIDAD DEL PROBLEMA = PUNTAJE TOTAL / PESO TOTAL**
			7.666
ASUNTOS DE VIABILIDAD DE DESPLIEGUE			
Puntaje =	**Peso ***	**Valor**	**Asunto**
56	7	8	El sistema puede ser introducido fácilmente
54	9	6	El sistema puede ser mantenido
42	7	6	El sistema no tiene una ruta critica
72	9	8	El sistema puede ser integrado con recursos existentes
56	7	8	Entrenamiento disponible
Puntaje Total: 280	**Peso Total: 39**		**VIABILIDAD DEL PROBLEMA = PUNTAJE TOTAL / PESO TOTAL**
			7.179

Luego de realizar el estudio de viabilidad para cada categoría resulto en los siguientes puntajes:

CATEGORÍA	PUNTAJE TOTAL	PESO TOTAL
Problema	697	91
Personal	665	81
Despliegue	280	39
Total	1642	211

VIABILIDAD DEL PROYECTO = PUNTAJE TOTAL / PESO TOTAL

VIABILIDAD DEL PROYECTO = 7.78

Anexo 2

Parámetros estadísticos de los modelos de regresión lineal múltiple.

Dim	R^2	Q^2_{loo}	Q^2_{boot}	$a(Q^2)$	Q^2_{ext}	$SDEP_{ext}$	Modelos
2	0.5905	0.5596	0. 5609	-0.175	0.5966	1.897	**ln_kf** = 0.05378(±0.1204)CA_I50_Q_M35_NS1_T_KA_Z1_MCoMPAs + 0.00423(±0.03505)CA_I50_B_M35_NS1_T_KA_Z2-GCP1_MCoMPAs + 17.78938(±1.47792)
3	0.7004	0.6697	0.6657	-0.158	0.5926	1.907	**ln_kf** = 0.02316(±0.08675)CA_Q2_F_M34_SS1_T_KA_EPS_MCoMPAs + 0.00455(±0.02263)CA_I50_B_M35_NS1_T_KA_Z2-GCP1_MCoMPAs + 0.18834(±0.79698)CA_I50_B_M35_NS1_T_KA_Z3-PTT_MCoMPAs + 55.2354(±9.72145)
4	0.7632	0.6697	0.7193	-0.143	0.4128	2.289	**ln_kf** = 0.22605(±1.07765)CA_I50_B_M35_MP1_T_KA_ISA-KDS_MCoMPAs + 0.00703(±0.03039)CA_I50_B_M35_NS1_T_KA_Z3-GCP1_MCoMPAs + 0.77786(±1.98047)CA_S_B_M33_NS1_T_KA_MM-ECI_MCoMPAs + 0.12825(±0.58616)CA_I50_B_M34_NS1_T_KA_Z1-PTT_MCoMPAs + 14.83484(±2.0424)
5	0.7888	0.7540	0.7450	-0.125	0.2030	2.667	**ln_kf** = 0.24314(±0.67468)CA_S_Q_M35_NS1_T_KA_Z3_MCoMPAs + 0.22132(±0.91044)CA_I50_B_M35_MP1_T_KA_ISA-KDS_MCoMPAs + 0.00665(±0.02989)CA_I50_B_M35_NS1_T_KA_Z3-GCP1_MCoMPAs + 0.11907(±0.56999)CA_I50_B_M34_NS1_T_KA_Z1-PTT_MCoMPAs + 0.01171(±0.03874)CA_N1_B_M34_MP1_T_KA_MV-HWS_MCoMPAs + 1.85157(±1304191)
6	0.7998	0.7583	0.7460	-0.106	0.1485	2.757	**ln_kf** = 0.01254(±0.02831)CA_N1_B_M34_SS1_T_KA_Z3-PTT_MCoMPAs + 0.49664(±CA_I50_F_M34_NS1_T_KA_PAH_MCoMPAs) + 0.24523(±CA_I50_B_M35_MP1_T_KA_ISA-KDS_MCoMPAs) + 0.11397(±0.59656)CA_I50_B_M34_NS1_T_KA_Z1-PTT_MCoMPAs + 0.01176(±0.03115)CA_N1_B_M34_MP1_T_KA_MV-HWS_MCoMPAs + 0.16662(±0.56297)CA_I50_B_M35_NS1_T_KA_Z3-PTT_MCoMPAs + 18.46635(±2.23673)
7	0.8259	0.7855	0.7734	-0.118	0.1112	2.816	**ln_kf** = 0.02186(±0.06673)CA_I50_B_M33_SS1_T_KA_ISA-PBS_MCoMPAs + 0.46727(±1.48369)CA_I50_F_M34_NS1_T_KA_PAH_MCoMPAs + 0.02493(±0.06048)CA_I50_B_M34_SS1_T_KA_MV-PBS_MCoMPAs + 0.01704(±0.07506)CA_N1_B_M33_SS1_T_KA_Z3-ECI_MCoMPAs + 0.12248(±0.47695)CA_I50_B_M34_NS1_T_KA_Z1-PTT_MCoMPAs + 0.01077(±0.04482)CA_N1_B_M34_MP1_T_KA_MV-HWS_MCoMPAs + 0.1346(±0.85586)CA_I50_B_M35_NS1_T_KA_Z3-PTT_MCoMPAs + 2.7342(±23.15753)