Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Estudio, selección e implementación de algoritmos de selección con ranking para mejorar la clasificación de interacciones de proteínas utilizando energías

Tesis presentada por Víctor Ignacio Gutiérrez Contreras

para obtener el grado de Ingeniero Civil en Informática

Dirigida por Tatiana Gutiérrez Bunster

2016

Resumen

"*Sin sacrificio no hay victoria...*"

Archibald Amundsen Witwicky

La Interacciones Proteína-Proteína (IPP) tienen un papel muy importante para poder descubrir la función que cumple una proteína como también en diversos procesos celulares y biológicos. Dada esta importancia existe una gran cantidad de investigaciones sobre las Interacciones Proteína-Proteína las cuales han dejado mucha información acumulada en las bases de datos. La ciencia de la computación ha atraído mucho la atención en estas últimas décadas por su capacidad de procesar grandes cantidades de datos las cuales pueden ser aplicadas como técnicas para entender el funcionamiento de las Interacciones Proteína-Proteína y la clasificación de los resultados experimentales. Para clasificar las Interacciones Proteína-Proteína se ha demostrado en los últimos años que es necesario poner gran atención sobre la zona de interacción en donde se producen diferentes tipos de características energéticas que pueden contribuir para la clasificación de complejos proteicos. En esta investigación se trabaja en base a las propiedades de características energéticas sobre complejos de proteínas que fueron previamente clasificados según su tiempo de duración en la interacción, además de complementar investigaciones anteriores desde donde se obtiene un conjunto de características energéticas representadas a través de matrices sobre las cuales se utilizó algoritmos de selección de características para obtener como salida un conjunto final de estas características energéticas ordenado por relevancia según un criterio de evaluación. Los datos de las matrices fueron obtenidos a través de la aplicación FastContact en donde se obtienen las características energéticas de la superficie de interacción entre proteínas. Para poder medir los resultados aplicando los algoritmos de selección de características se utilizó la aplicación Weka la cual es usada para obtener el nivel de precisión de la clasificación de las clases.

Palabras Clave — Interacciones Proteína-Proteína, Proteína, Selección de características, Zona de interacción, Características energéticas.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las Interacciones Proteína-Proteína tiene un papel importante en los procesos celulares tales como transducción de señales[1], transporte a través de membranas [2], metabolismo celular [3]y en otros ámbitos biológicos. Es por esto que muchos investigadores se han motivado en intentar comprender estas interacciones. Existen diferentes métodos para poder determinar el tipo de interacción, algunos de los cuales son realizados en laboratorio inclusive dentro de organismos vivos, pero esto conlleva a un gasto alto de recursos, como alternativa han surgido variados esfuerzo para aplicar la ciencia de la computación como herramienta que permita obtener conocimiento sobre la interacción entre proteínas.

Existe una gran cantidad de datos almacenados de complejos proteicos los cuales se pueden encontrar en diferentes bases de datos de proteínas, estos datos pueden ser utilizados para poder generar estudios sobre la zona de interacción entre las proteínas. En diferentes áreas nace la necesidad de estudiar y analizar una gran cantidad de datos, como pueden ser las bases de datos de proteínas, en estos casos se necesita utilizar técnicas de la ciencia de la computación relacionadas con el reconocimientos de patrones para poder extraer conocimientos desde esta

[1]La transducción de señales a nivel celular se refiere al movimiento de señales desde fuera de la célula a su interior.

[2]El transporte de membrana se refiere al conjunto de mecanismos que regulan el paso de solutos, tales como pequeñas moléculas, a través de membranas plasmáticas.

[3]El metabolismo celular es el conjunto de reacciones químicas que se producen en el interior de las células de un organismo, mediante las cuales los nutrientes que llegan a ellas desde el exterior se transforman.

gran cantidad de datos. Dentro del reconocimiento de patrones existen diferentes etapas las cuales pueden ser diferenciadas por su función, primero se necesita obtener los datos de una fuente como son las base de datos de proteínas, luego se debe procesar estos datos y clasificarlos dependiendo sus características y finalmente se debe evaluar la clasificación.

En las Interacciones Proteína-Proteína existen diferentes metodologías para estudiar su zona de interacción, pocas de ellas utilizan las características energéticas como discriminante entre un tipo de interacción u otra. Esta investigación se centra en utilizar estas características energéticas como criterio de discriminación entre complejos que se diferencian en el tiempo de duración en la interacción. Para este propósito se necesitan algoritmos capaces de seleccionar las características energéticas que aporten mejor información correcta y que disminuyan la tasa de error en la clasificación. Hay un extenso número de algoritmo de selección de características encontrado en la bibliografía, los cuales se diferencia en su dirección de búsqueda, su criterio de evaluación sobre los subconjuntos de características y si entregan un subconjunto óptimo o sub-óptimo.

1.1. Objetivos

Este trabajo de investigación tiene como objetivo el estudio de algoritmos de selección de características, que ayude a seleccionar las características que sean más relevantes en su grupo para lograr que la clasificación de las clases de complejos de Interacciones Proteína-Proteína permanentes y transitorias sea de una precisión[4] mayor.

Para lograr esto se deben buscar y estudiar los algoritmos de selección que permitan seleccionar las características energéticas de las proteínas. Una vez realizado el estudio previo se debe analizar qué algoritmo puede ser implementado en el tiempo del transcurso de la investigación, se debe analizar su comportamiento en cuanto a al complejidad en el tiempo de ejecución y que tipo de subconjunto (óptimo o sub-óptimo) entrega para su comprobación con estudios anteriores.

Se implementará los algoritmos elegidos en lenguaje de programación Python, obteniendo resultados del subconjunto seleccionado por los algoritmos. Finalmente se debe estudiar la precisión en la clasificación de los subconjuntos de características entregadas por los algoritmos, para lo cual se utiliza la aplicación Weka [16].

[4]Se denomina precisión al resultado de dividir el número de clasificaciones correctas por el número total de muestras.

1.2. Organización

- Capítulo 1: **Proteínas**.

  Se realiza una introducción al tema de proteínas, para poder entender el manejo de los datos utilizados y la importancia de esta investigación desde el área biológica.

- Capítulo 2: **Patrones**.

  Se introduce al lector sobre los conocimientos básicos sobre los reconocimientos de patrones para situar en el contexto en donde se localiza la selección de características.

- Capítulo 3: **Selección de Características**.

  Se realiza un estudio sobre los algoritmos de selección de características, presentando las diferencias que existen según el enfoque en el cual son creados.

- Capítulo 4: **Estudio del Problema**.

  Se presenta la metodología con la cual se abordo el problema, la obtención de los datos y los algoritmos a utilizar para los siguientes capítulos.

- Capítulo 5: **Implementación y Aplicación**.

  Se realiza una breve introducción sobre la implementación de los algoritmos desarrollados y la aplicación sobre las características obtenidas del capítulo anterior.

- Capítulo 6: **Resultados**.

  Se presentan los resultados obtenidos desde la clasificación de diferentes conjuntos de datos obtenidos a través de los algoritmo de selección de características, además de la interpretación de los resultados.

- Capítulo 7: **Conclusiones**.

  Finalmente se exponen las conclusiones de esta investigación, en la cual se presentan los métodos estudiados presentando las fortalezas y debilidades. Además de mencionar trabajos futuros sobre la misma linea de la investigación.

Capítulo 2

Proteínas

Desde la primera vez que fueron descritas formalmente las proteínas en el año 1838 por el químico sueco Jöns Jakob Berzelius, estas han sido ampliamente estudiada hasta nuestros días ya que cumplen un papel fundamental en cada organismo viviente en la tierra. La gran cantidad de información recabada por estos estudios hizo indispensable la utilización de la ciencia de la computación para procesar la información de forma más rápida y eficaz. Actualmente el estudio de datos biológicos a través de la informática es conocida como BioInformática, una definición más elaborada en [26] señala que la BioInformática es:

> "La conceptualización de la biología en términos de moléculas (en el sentido químico físico) y la aplicación de técnicas informáticas (derivadas desde las disciplinas aplicadas como la matemática, ciencias de la computación y estadística) para comprender y organizar la información asociada con las moléculas, a gran escala. En resumen, BioInformática es un sistema de manejo de la información para la biología molecular que posee muchas aplicaciones prácticas".
>
> (Luscombe, Greenbaum y Gerstein, 2001)

Existen variadas bases de datos en las cuales se almacena la información existente sobre las proteínas, estas bases de datos se diferencias por la forma en que obtienen la información utilizando diferentes métodos e investigaciones. Una de las base de datos pioneras establecida en

el año 1971 es la reconocida Protein Data Bank (PDB) [2] en la cual actualmente se encuentran almacenadas sobre 120.000 estructuras proteicas y esta cifra va aumentando cada año.

Antes de poder utilizar la información almacenada en las bases de datos de proteínas a través de la informática, es necesario introducir la importancia de las proteínas a nivel biológico, es por esto que en este capítulo se presentarán dos secciones en la primera se describirá la composición de una proteína y en la segunda se describirá las Interacciones Proteína-Proteína.

2.1. Proteínas

Las proteínas cumplen diferentes funciones especificas que permite que la célula mantenga su estructura, se defienda de agentes externos, pueda transportar Oxígeno, entre otras funciones que hacen de las proteínas unas de las biomoléculas más versátiles y diversas. La composición de las proteínas es básicamente de Carbono, Hidrógeno, Oxígeno y Nitrógeno. Pueden además contener Azufre y en algunos tipos de proteínas contienen Fósforo, Hierro, Magnesio y Cobre entre otros elementos.

Las proteínas se pueden representar a través de aminoácidos que se pliegan adquiriendo una estructura tridimensional. Existen 20 aminoácidos estándar los cuales se pueden unir para formar una proteína, en general la cantidad de combinaciones de aminoácidos se ve descrita como 20^n donde n es la cantidad de aminoácidos que componen una proteína. Pero cada una de estas combinaciones no generan necesariamente una proteína útil, en [28] se estima que alrededor de 650.000 a 2.000.000 millones de secuencias de proteínas es realmente generada por los seres vivos.

Los aminoácidos para poder unirse y conformar una proteína deben hacerlo a través de enlaces peptídicos dando como resultado una cadena de aminoácidos. La unión de un bajo número de aminoácidos da lugar a un péptido; si el número de aminoácidos no es mayor a 10 se denomina oligopéptido, si es superior a 10 es polipéptido y si el número de aminoácidos es superior a 50 aminoácidos se refiere a una proteína. Por ejemplo, según [3] la insulina es una molécula proteica con 51 aminoácidos mientras que la apolipoproteína B (una proteína transportadora de colesterol) contiene 4.536 aminoácidos, que representa la cadena individual de aminoácidos más grande conocida hasta la fecha.

Para entender la composición de un aminoácido y su unión a través de enlaces peptídicos en la siguiente sección 2.1.1 se explicará el funcionamiento de manera ilustrativa a través de imágenes. En la sección 2.1.2 se explicarán las diferentes estructuras que existen para poder estudiar las proteínas y finalmente en la sección 2.1.3 se describirán las funciones que cumplen las proteínas a nivel biológico.

2.1.1. Aminoácidos

Los aminoácidos son la estructura fundamental de una proteína, se puede definir como una molécula orgánica que se caracteriza por poseer un grupo carboxilo, un grupo amino, unido a un átomo de Carbono, uno de Hidrógeno y a un grupo que varia según el aminoácido del que se trate, llamado residuo (R), su estructura se puede ver en la Figura 2.1.

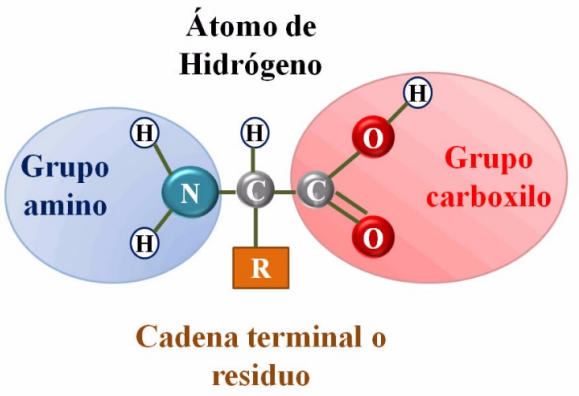


Figura 2.1: Composición Aminoácido [37].

Cuando dos aminoácidos A y B se combinan en una reacción de condensación[5] lo hacen a través del grupo carboxilo de A y el grupo amino de B, liberándose una molécula de agua (H_2O) y formándose un enlace, denominado enlace peptídico. En la Figura 2.2 se puede ver representada esta transición entre los aminoácidos A y B. La unión entre dos aminoácidos forma

[5]La reacción de condensación es una reacción química en la que dos moléculas se combinan para formar una molécula más grande, junto con la perdida de una molécula pequeña, que en la mayoría de los casos es una molécula de agua.

un dipéptido, si se une un tercer aminoácido se forma un tripéptido y así sucesivamente hasta formar un polipéptido.

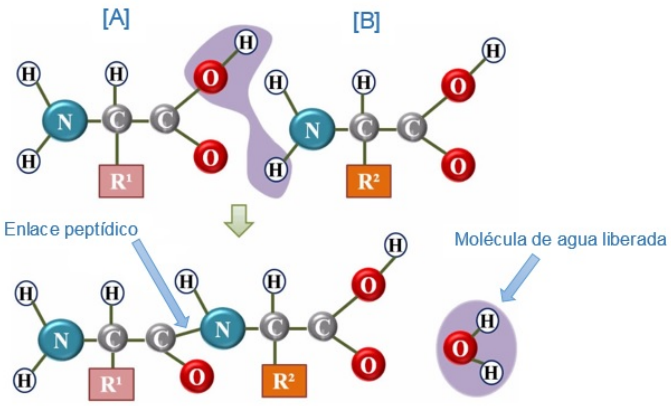


Figura 2.2: Enlace peptídico [37].

2.1.2. Estructuras de las proteínas

Las proteínas son biomoléculas muy complejas de comprender, su composición se encuentra estructurada como una cadena aminoácidos donde su longitud y secuencia no es al azar, es por eso que para estudiar una proteína se diferencian en cuatro diferentes estructuras. Cuando las proteínas tiene una composición determinada de aminoácidos y están ordenados en una determinada secuencia, dicha secuencia es conocida como Estructura primaria. Si la secuencia lineal de los aminoácidos cambia, además dependiendo de cómo se encuentren enlazados y cómo se plieguen se conoce como Estructura secundaria. La forma en que se pliega las cadenas de aminoácidos en el espacio es la Estructura terciaria. Las proteínas en general no están compuestas por sólo una cadena de aminoácidos, cuando se agrupan varias cadenas de aminoácidos se habla de Estructura cuaternaria. En la Figura 2.3 se puede apreciar las diferentes estructuras que se utilizan para estudiar las proteínas.

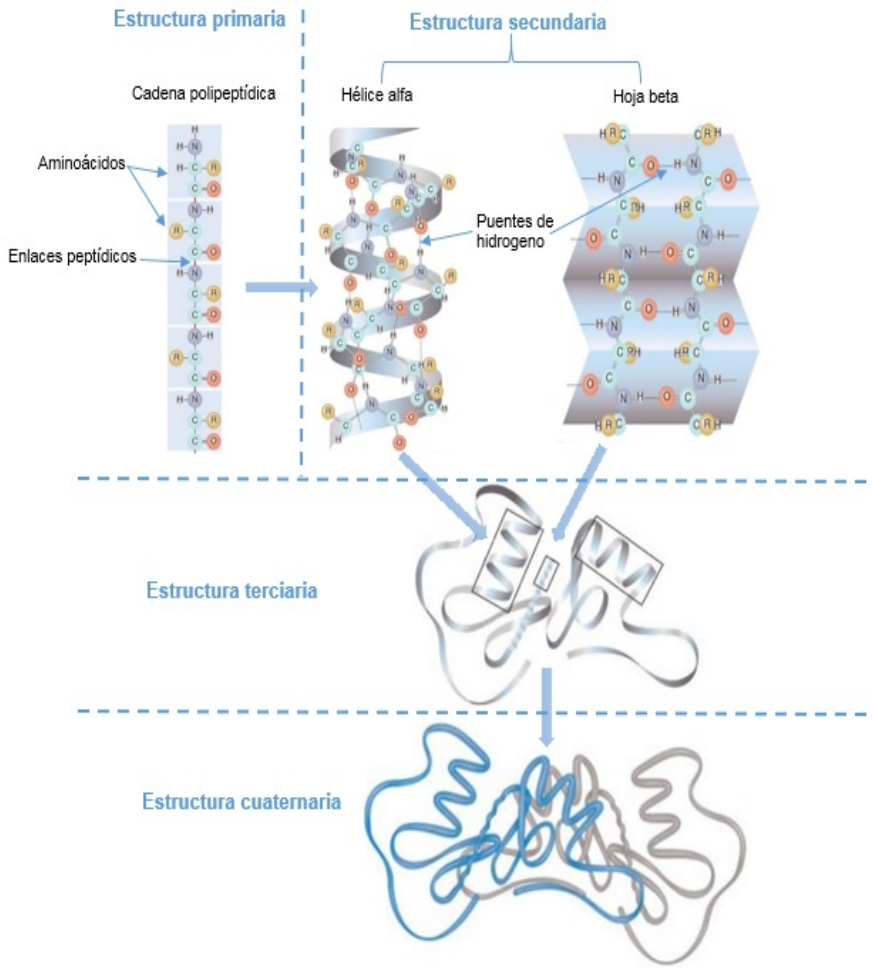


Figura 2.3: Estructura de las Proteínas [10].

1. Estructura Primaria: La estructura primaria se ve determinada por la secuencias de aminoácidos según el número de aminoácidos presentes y el orden en que estos se enlazan formando la cadena polipeptídica. La estructura primaria de una proteína es la que determina la función que cumplirá.
2. Estructura Secundaria: La estructura secundaria de una proteína es el modo en que los aminoácidos se van plegando por la formación de puentes de hidrógenos en el espacio. Es el primer nivel de plegamiento en que los aminoácidos se disponen de un modo ordenado y repetitivo siguiendo una dirección. Existen dos tipos de estructuras secundarias denominadas

Hélice alfa y Hoja Beta

3. Estructura Terciaria: La estructura terciaria es la disposición tridimensional de todos los átomos que componen la proteína estos se mantiene estables debido a la existencia de enlaces entre los radicales R de los aminoácidos, como los puentes de hidrógeno, puentes eléctricos, interacciones hidrófobas y puente disulfuro.
4. Estructura Cuaternaria: La estructura cuaternaria se refiere a las proteínas que están formadas por varias cadenas polipeptídicas, conocidas como proteínas oligomericas. Las proteínas oligomericas están formadas por un número variable de protomeros o subunidades.

2.1.3. Funciones de las proteínas

Las proteínas cumplen varias funciones de gran relevancia y están diferenciadas según su comportamiento. Cada proteína cumple con una función específica, algunas de estas funciones son descritas brevemente a continuación:

1. Estructural: La función estructural de las proteínas tiene relación con la formación de tejidos de sostén y relleno que confieren elasticidad y resistencia a órganos y tejidos.
2. Enzimática: Las proteínas que tienen como función enzimática son las más numerosas, en su función como enzima hacen uso de la propiedad de poder interactuar con diversas moléculas.
3. Hormonal: Algunas de las proteínas que cumplen esta función son la insulina y el glucagón que regulan los niveles de la sangre. Otras como la hormona del crecimiento que se involucra con el crecimiento de lo tejidos, músculos, el mantenimiento y reparación del sistema inmunológico. Básicamente son las proteínas que cumplen con una función regulatoria de procesos metabólicos.
4. Defensiva: Las proteínas son capaces de crear anticuerpos y regular factores contra agentes externos como infecciones. Algunos ejemplos son el fibrinógeno y la trombina que contribuyen a la formación coágulos de sangre para evitar las hemorragias
5. Transporte: Estas son transportadoras del oxígeno en la sangre en los organismos vertebrados, por ejemplo la hemoglobina y la mioglobina. También existen proteínas que

transportan electrones como es el caso de el citocromos.

6. Reserva: Algunas proteínas que cumplen con esta función son la ovoalbúmina de la clara de huevo, la gliadina del grano de trigo y la hordeina de la cebada, que constituyen la reserva de aminoácidos para el desarrollo del embrión.
7. Reguladoras: Existen proteínas que pueden regular la expresión de ciertos genes y otras que pueden regular la división celular como es el caso de la ciclina.
8. Función Homeostática: Estas proteínas funcionan como amortiguadores para mantener el equilibrio osmótico y el pH del medio interno.

2.2. Interacciones Proteína-Proteína (IPP)

Las Interacciones Proteína-Proteína (IPP) se refiere al contacto físico entre dos o más proteínas como resultados de eventos químicos y/o fuerzas electrostáticas, a las proteínas que conforman la unión de la interacción se le denominan complejo, en la Figura 2.4(b) se puede apreciar la zona de interacción entre dos Proteína. Las IPP juega un papel crítico para una amplia gama de procesos biológicos tales como transducción de señales, transporte a través de membranas, metabolismo celular, entre otros. En [1] se menciona que sobre el 80 % de las Proteínas no actúan por si solas, es por esto que es necesario estudiar el comportamiento de las Interacciones entre Proteínas para poder entender su función.

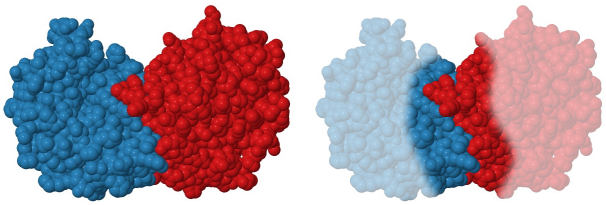
Algunas de las propiedades importantes de las IPP fueron definidas por Phizicky y Fields [36] en donde las IPP pueden: (1) Modificar las propiedades cinéticas de las enzimas[6]; (2) Actuar como un mecanismo para permitir la canalización de sustratos; (3) Construir un nuevo sitio de unión para moléculas pequeñas; (4) Desactivar o suprimir una proteína; (5) Cambiar la especificidad[7] de una proteína.

Debido a la gran importancia que tienen las IPP en prácticamente todos los procesos celulares, han surgido diferentes estudios los cuales han dejado una gran cantidad de información almacenadas en diferentes bases de datos (BD). Algunas de estas BD son DIP (Database of Interacting

[6]La cinética enzimática es la disciplina que estudia la velocidad en las reacciones químicas en las que intervienen enzimas.

[7]La especificidad de las proteínas indica que cada una de ellas lleva a cabo una determinada función.

Proteins) y BIND (Biomolecular Interaction Network Database), las cuales están descritas en [42]. El poder extraer conocimiento desde estas BD puede contribuir a mejorar el conocimiento de enfermedades y poder proporcionar nuevos enfoques para tratamientos terapéuticos.



(a) Interacción Proteína-Proteína (b) Zona de Interacción

Figura 2.4: Representación Zona de interacción (PDB ID: 1TIM) [2].

En la Figura 2.4 se observa la IPP y la zona de interacción, esta imagen fue tomada desde la base de datos Protein Data Bank (PDB) [2] y visualizada utilizando JSMOL desde la misma plataforma web. En la Figura 2.4(b) se puede ver que ambas proteínas están en contacto, a esta sección se le denomina zona de interacción, esta zona posee propiedades diferentes del resto de la superficie lo que permite la interacción entre proteínas.

2.2.1. Clasificación de los métodos de detección de IPP

Los métodos de detección de IPP se pueden categorizar en tres tipos, *in vitro*, *in vivo* y *in silico*. En los métodos *in vitro* el procedimiento se lleva a cabo en un ambiente controlado fuera de un organismo vivo, pertenecientes a este método se pueden mencionar Coinmunoprecipitación, Cristalografía de rayos X y Espectroscopia de resonancia magnética nuclear (RMN). Los métodos denominados *in vivo* se llevan a cabo en el propio organismo vivo, a este método pertenecen métodos como el Sistema de dos híbridos (Yeast Two Hybrid - Y2H). Por último el método *in silico*, se realiza a través de un computador o a través de simulación por computador, se puede mencionar enfoques basados en la estructura y en la expresión génica. En la Tabla 2.1 se muestran algunos de los métodos de detección de IPP su descripción se encuentra en el Apéndice A.

Enfoque	Técnica
In vitro	Cristalografía de rayos X
	Coinmunoprecipitación
	Espectroscopía de resonancia magnética nuclear
In vivo	Sistema de dos híbridos (Y2H)
In silico	Enfoques basados en la expresión génica
	Enfoques basados en la estructura

Tabla 2.1: Descripción de métodos de detección de IPP. Adapatada de [40]

2.2.2. Tipos de interacción

Casi todos los procesos celulares requieren que las proteínas interactúen con otras proteínas, estas interacciones se clasifican en diferentes tipos dependiendo de diferentes factores que ocurren en la interacción, un estudio realizado por Nooren y Thornton [33] clasifica los complejos como:

1. Complejos Homo y hetero-oligoméricos: Un complejo que consta de solamente proteínas idénticas se considera que es un homo-oligómeros, por el lado contrario un complejo formado de diferentes proteínas se define como hetero-oligomeros.
2. Complejos obligados y no obligados: En un complejo obligado las proteínas no forman estructuras estables cuando se encuentran separadas, es decir, que sólo tienen un funcionamiento en conjunto. Mientras que las no obligadas pueden tener una existencia independiente, es decir, pueden mantenerse estables de forma separada.
3. Complejos transitorios y permanentes: Las IPP se pueden diferenciar basándose en el tiempo de vida de la interacción. Las interacciones permanentes son muy estables, lo que implica que la vida del complejo se mantenga por más tiempo, por otro lado las interacciones transitorias se asocian y disocian continuamente *in vivo*.

Desde los últimos años se ha estado trabajando para que los datos obtenidos desde las IPP sean de mejor calidad, algunos de estos métodos experimentales han sido nombrados en la sección 2.2.1. Se han realizado esfuerzos por tratar de investigar las IPP utilizando técnicas informáticas en conjunto con análisis computacional para poder comprender las funciones e interacciones de las proteínas aún no exploradas. Estas investigaciones en general utilizan la información estructural

de las proteínas que se encuentran almacenadas en las bases de datos pero aún no existe un método estandarizado que determine qué técnica o método es el correcto para poder comprender las funciones o iteraciones entre proteínas.

En los siguientes capítulos se tratarán temas relacionados con el reconocimiento de patrones en el cual se busca extraer información desde grandes conjuntos como son las bases de datos de proteínas. Además se centrará en estudiar los algoritmos de selección de características con los que se pretende conseguir un conjunto de datos que puedan ser usados para la clasificación de las Interacciones Proteína-Proteína.

Capítulo 3

Patrones

El proceso de extraer conocimiento desde los datos se realizaba de forma manual quedando el análisis y la interpretación en función de los conocimientos del especialista. Esta forma de procesar los datos traía consigo un proceso lento, caro y altamente subjetivo. En muchos casos la cantidad de datos desborda la capacidad humana por lo cual el análisis manual se transforma en una decisión de intuición en base a la experiencia del especialista. Es por esto que hace más de tres décadas se está implementando y utilizando algoritmos para la extracción de patrones para grandes conjuntos de datos. A finales de la década de los 80 al conjunto de métodos matemáticos y técnicas software para la búsqueda de tendencias, se denominó Minería de Datos (Data Mining[8] - DM). El nombre de Minería de datos se relaciona inmediatamente con buscar información que resulte valiosa en una mina de datos, esta información es como encontrar una veta de metales preciosos. Una definición entregada por Decker y Focardi [7] señala que:

> "La Minería de datos es una metodología de resolución de problemas que encuentra una descripción lógica o matemática, de naturaleza compleja, de los patrones y regularidades en un conjunto de datos".
>
> (Decker y Focardi, 1995)

[8] Data mining is a problem solving methodology that finds a logical or mathematical description, eventually of a complex nature, of patterns and regularities in a set of data. (Decker y Focardi, 1995).

La Minería de datos debe ofrecer la posibilidad de generar automáticamente nuevas hipótesis, al contrario de los métodos estadísticos tradicionales que se utilizan para verificar o desaprobar una hipótesis previa. El objetivo de la Minería de datos es extraer conocimiento eficiente y eficaz a partir de los datos.

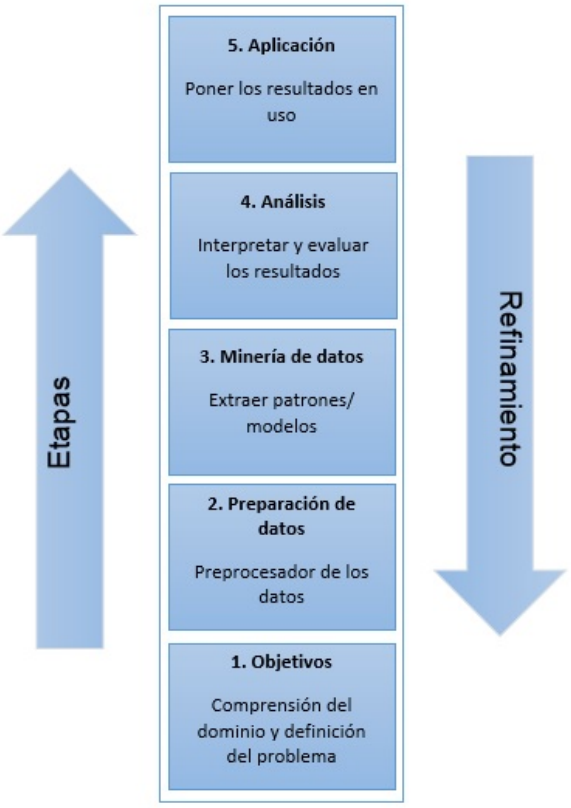


Figura 3.1: Esquema del "Descubrimiento de Conocimiento en Bases de Datos"(Knowledge Discovery in Databases - KDD) [41].

Como se puede observar en la Figura 3.1 la Minería de datos (etapa 3) necesita complementarse con una adecuada preparación de los datos previa al proceso de minería (etapa 2) y un análisis posterior de los resultados obtenidos (etapa 4). Entonces se deriva que la Minería de datos es una parte de un esquema más amplio conocido como Descubrimiento de Conocimiento en Bases de Datos (KDD[9]), el cual es el proceso completo de extraer conocimiento a partir de las bases de datos. Una definición entregada por Fayyad et al. [11] señala que:

> "El Descubrimiento de Conocimiento en Bases de Datos describe un proceso no trivial de identificación valida, nueva, potencialmente útil y por último define patrones comprensibles en los datos".
>
> (Fayyad et al, 1996)

Otra área de interés para la investigación y extracción del conocimiento es el reconocimiento de patrones. Esta es una rama de la Inteligencia artificial, conocida como aprendizaje automático (Machine Learning), que se ocupa para desarrollar técnicas capaces de aprender, es decir, extraer de forma automática conocimiento a través de información no estructurada suministrada en forma de muestras. Esto se logra al comparar el conjunto de prueba con el conjunto de entrenamiento previo.

El aprendizaje automático tiene una variada gama de aplicaciones en el procesamiento de la información, incluyendo motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, estudio y clasificación de cromosomas, reconocimientos de genes en secuencias, además de las áreas agrícolas, industriales, astronómicas, en las cuales se realizan procesamiento de imágenes, señales sísmicas, radar, diagnóstico de enfermedades, fallos en maquinarias reconocimiento del habla y del lenguaje escrito, juegos, robótica entre otros [14].

Este capítulo se enfocará en determinar en que área se sitúa la selección de características para extraer conocimiento sobre un espacio de características.

[9]KDD is the non trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. (Fayyad et al, 1996).

3.1. Enfoques de Reconocimientos de patrones

El reconocimiento de patrones presenta diferentes enfoques, los cuales no son necesariamente independientes y a veces el mismo método de reconocimiento de patrones puede tener diferentes interpretaciones. Una breve descripción y comparación de estos enfoques se muestra en la Tabla 3.1.

3.1.1. Comparación de plantillas (Template Matching)

El enfoque denominado Comparación de plantillas es uno de los más sencillos y está orientado a determinar el grado de similitud entre dos entidades del mismo tipo, que pueden ser puntos, curvas, muestras, píxeles u otras formas. En [21] se dice que para lograr el grado de similitud se debe disponer de un prototipo asociado al patrón a reconocer del cual se quiere aprender a partir de los datos de entrenamiento.

3.1.2. Reconocimiento Estadístico de Patrones

En el enfoque Estadístico, cada patrón está representado en términos de x características, constituyendo un punto en el espacio de d dimensiones. El objetivo es seleccionar aquellas características que permitan que los patrones que pertenezcan a distintas categorías ocupen regiones compactas y disjuntas en el espacio de características, de forma tal de poder separar los elementos de cada clase adecuadamente. Para ésto, y en base a un conjunto de patrones de entrenamiento, se establecen límites de decisión en el espacio de características, por ejemplo, en base a las distribuciones de probabilidades de los patrones de cada clase [21].

3.1.3. Reconocimiento Sintáctico de Patrones

En el enfoque Sintáctico, se establece una analogía formal entre las estructura de los patrones y la sintaxis del lenguaje. Los patrones se consideran como estructuras u oraciones del lenguaje, mientras que las primitivas o subpatrones elementales constituyen el alfabeto, de forma tal que estas estructuras y oraciones son generadas de acuerdo a una gramática. Así, un conjunto de patrones complejos se puede describir utilizando un pequeño número de primitivas y reglas

gramaticales. La gramática asociada a cada clase se infiere del conjunto de patrones de entrenamiento. Según [25] el enfoque sintáctico presenta algunas dificultades como, por ejemplo, la necesidad de utilizar grandes conjunto de datos y estar asociado a altos costos computacionales. Hay aplicaciones con este tipo de reconocimiento usadas en la biología molecular para el análisis de secuencias de proteínas [22].

3.1.4. Redes Neuronales

Las Redes Neuronales pueden ser vista como un masivo sistema de computación en paralelo de un largo número de simples procesos con muchas interconexiones. Estas tienen la característica de poder aprender relaciones no lineales complejas entre valores de entrada y salida, entrenarse de forma automática mediante muestras y poder aprender a partir de grandes base de datos, adaptándose a los datos lo cual presenta un muy buen rendimiento frente a datos con ruido [25].

Enfoque	Representación	Función de reconocimiento	Criterios Típicos
Template matching	Muestras, Píxeles y Curvas	Correlación, Medida de distancia	El error de clasificación
Estadístico	Características	Función discriminante	El error de clasificación
Sintáctico	Primitivas	Reglas, Gramática	El error de aceptación
Redes Neuronales	Muestras, Píxeles y Características	Función de la Red	Error cuadrático medio

Tabla 3.1: Enfoque de Reconocimiento de patrones [21].

3.2. Etapas

El reconocimiento de patrones está compuesto por diferentes etapas que incluyen un sensor que recoja los elementos o datos a ser clasificados, un mecanismo de reducción de dimensionalidad para evitar la información redundante e irrelevante que pueda causar ruido en la etapa final que es la clasificación, la cual se basa en las características extraídas. El proceso de reconocimiento de patrones no está claramente estandarizado y es por eso que para esta investigación se seguirá el siguiente enfoque mostrado en la Figura 3.2.

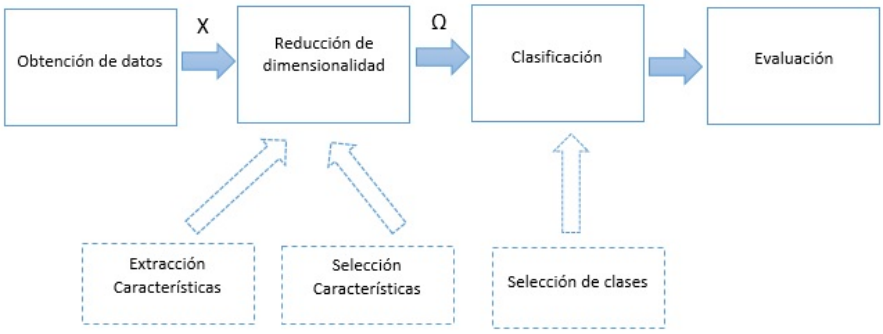


Figura 3.2: Proceso de reconocimiento de patrones.

Como se puede ver en la Figura 3.2, la entrada al sistema de reconocimiento es un conjunto de medidas (o muestras tomadas en la obtención de los datos) realizadas sobre el vector X, denominado "vector patrón", dicho vector contiene toda la información disponible de la señal. Luego desde el vector X se crea un nuevo conjunto de características, en donde las características están estructuradas en forma de vector de características Ω. El clasificador opera sobre el vector de características Ω, con un conjunto de funciones denominadas "decisiones" o "funciones discriminantes" para obtener la clasificación.

3.2.1. Obtención de los datos

El flujo de datos que hay a nuestro alrededor es evidentemente alto. De todas las cosas se puede obtener datos y realizar análisis a partir de éstos, el problema es cómo obtener éstos datos. Años atrás era más difícil obtener datos de las cosas tales como magnitudes físicas o químicas, pero con la revolución de la tecnología este proceso se volvió más fácil y existen una variedad de instrumentos dedicados a este proceso tales como la cámara, el micrófono, termómetros, y un sinfín de otros. Estos instrumentos nos entregan variables tales como temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, etc.

3.2.2. Reducción de dimensionalidad

La finalidad de esta etapa es encontrar aquellas características que representan de mejor manera a cada tipo de objeto. Estas características deben ser entregadas de una manera clara para ser utilizadas [14]. Existen dos razones para reducir la dimensionalidad (el número de características) lo más pequeño posible; el costo de medición y la precisión de la clasificación. Es importante hacer una distinción entre la extracción de características y la selección de características. El término selección de características se refiere a los algoritmo que seleccionan las características más relevantes para crear un subconjunto desde el conjunto de entrada, en cambio la extracción de características hace referencia a la creación de nuevas características basado en transformaciones o combinaciones del conjunto original de entrada.

3.2.2.1. Extracción de características

Los métodos de extracción de características determinan un sub-espacio apropiado de dimensionalidad m (en forma lineal o no lineal) desde las características originales de dimensionilidad d ($m \leq d$). Las transformaciones lineales, tales como el análisis de componentes principales (ACP), análisis de factores, análisis discriminante lineal y la búsqueda de proyección han sido ampliamente utilizados en el reconocimiento de patrones para la extracción de características y la reducción de dimensionalidad. El extractor de características lineales más usado es el Análisis de Componentes Principales (Principal Component Analysis - PCA) [21] en el cual se representa el conjunto original con un número menor de características las que son construidas como combinaciones lineales de las originales. Su utilidad es :

1. Permitir representar óptimamente un espacio de una dimensión mas pequeña sobre un espacio general mayor.
2. Permitir transformar las variables originales facilitando la interpretación de los datos.

3.2.2.2. Selección de características

La selección de característica busca escoger un subconjunto mínimo de características en base a dos criterios [41]; (1) que la tasa de aciertos no descienda en base al conjunto original y (2) que la distribución de clase resultante, sea lo más representativa posible a la distribución de clase original dada todas las características.

La selección de características se puede ver como un conjunto de características de las cuales se selecciona un subconjunto de tamaño menor que conduzca a un error de clasificación más pequeño. En [21] dividen en dos la motivación por la cual la selección de características ha sido necesaria:

1. Fusión de sensores (Multisensor fusion): Las características, que son formadas a partir de diferentes fuentes de sensores, son concatenadas para formar un vector de características con un gran número de componentes.
2. La integración de múltiples modelos de datos: los datos del sensor se pueden obtener mediante diferentes enfoques, lo que implica que los parámetros del modelo sirven como características y los parámetros desde diferentes modelos pueden ser combinados para producir un vector con un gran número de características.

La selección de características se puede ver representada como sigue, se tiene Y como el conjunto de características con cardinalidad d y el número de características se representa con m y finalmente el subconjunto seleccionado X, en donde $X \subseteq Y$. El criterio de selección para el subconjunto X se representa por $J(X)$. Entonces se asume que un alto valor de J indica el mejor subconjunto de características. Más adelante, esta investigación entrará en detalle sobre la aplicación de la selección de características en el Capítulo 3.

3.2.3. Clasificación

Los métodos de clasificación son utilizados para poder aprender y extraer conocimiento desde un conjunto de datos el cual permite modelar ese conocimiento en una posterior aplicación en la toma de decisión. Formalmente se puede definir a un Método de clasificación como:

Definición "Sea C un conjunto de datos, el objetivo de la clasificación es aprender una función $L : X \rightarrow Y$, denominada clasificador, que presente la correspondencia existente en las muestras entre los vectores de entrada y el valor de salida correspondiente, es decir, para cada valor de x tenemos un único valor de Y, donde Y es nominal, es decir, puede tomar un conjunto de valor $y_1, y_2, y_3, ..., y_k$ denominados clases [41]".

Los datos de entrada del clasificador juegan un papel fundamental en el éxito del algoritmo de aprendizaje es por esto que se debe realizar una buena preparación de los datos etapa previa a la clasificación.

Una vez realizada la clasificación el modelo generado se puede representar de diferentes formas como representación proposicional, árboles de decisión, reglas de decisión, listas de decisión, reglas con excepciones, reglas jerárquicas de decisión, reglas difusas y probabilidades, éstas son las estructuras más utilizadas en la literatura.

3.2.3.1. Aprendizaje

En el aprendizaje automático se encuentran técnicas de clasificación que nos permiten agrupar muestras (características) de acuerdo a criterios o métodos, éstas técnicas pueden diferenciarse en clasificación supervisada y clasificación no supervisada.

1. Clasificación Supervisada: Este tipo de clasificación cuenta con un conocimiento a priori, es decir, para la tarea de clasificar un objeto dentro de una categoría o clase contamos con modelos ya clasificados (objetos agrupados que tienen características comunes). La clasificación supervisada se puede visualizar en dos etapas:
   a) La primera etapa tenemos un conjunto de entrenamiento (para el diseño del clasificador) y otro llamado prueba, estos servirán para construir un modelo para la clasificación.
   b) En la segunda etapa del proceso es clasificar los objetos en las que se desconoce la clase a las que pertenecen.
2. Clasificación no Supervisada: A diferencia de la clasificación supervisada no se cuenta con un conocimiento a priori, por lo cual no se tiene datos de entrenamiento disponible para la

tarea de clasificación. A la clasificación no supervisada se le suele llamar también clustering. En este tipo de clasificación se cuenta con objetos que tienen un conjunto de características, de las que no se sabe a que clase o categoría pertenece, entonces la finalidad es el descubrimiento de grupos de objetos cuyas características permitan separarse en diferentes clases.

3.2.3.2. Métodos de clasificación

Los métodos de clasificación se pueden diferenciar por su diseño de reglas y algoritmos de clasificación, en los cuales se pueden encontrar distribuciones probabilísticas, de contenido de información espacial y algebraica. A continuación se introducirán algunos conceptos de estos métodos los cuales fueron elegidos por su representatividad en diferentes estudios realizados en la literatura.

1. **Máquina de soporte de vectores (Support Vector Machine - SVM):** SVM es utilizado cuando los datos se encuentran separados en dos clases. Este clasificador busca encontrar el mejor hiperplano óptimo que separa todos los puntos de datos de una clase con respecto a la otra. Cuando se tiene un conjunto de datos que pertenecen a cada una de las dos clases como se puede ver en la Figura 3.3, se debe encontrar un hiperplano que separe las clases.

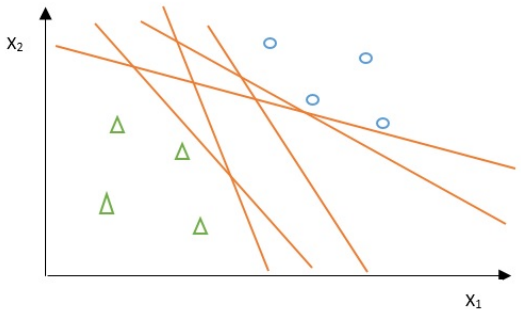


Figura 3.3: Máquinas de soporte de vectores conjunto de clases. Esta Figura es demostrativa en 2D, las líneas y puntos están en el plano cartesiano los cuales representan a los hiperplanos y vectores en un espacio dimensional más alto.

En la Figura 3.3 se ven diferentes líneas que podrían ofrecer una solución, el criterio es considerar que las líneas son malas u ofrecen una mala solución al problema si pasan cercanamente a los puntos de datos, ya que como se señala en [34] serán más sensibles al ruido y no va a generalizar correctamente. Entonces hay que buscar la línea que pasa equilibradamente más lejos entre todos los puntos.

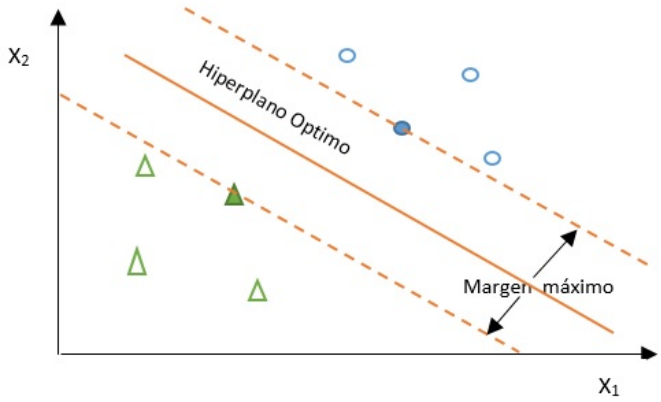


Figura 3.4: Máquinas de soporte de vectores Hiperplano.

El mejor hiperplano sería el que tiene el mayor margen entre las dos clases. El margen se refiere a la anchura o distancia que existe de forma paralela al hiperplano y que no existen puntos de datos en su interior. Los vectores de soporte son los datos que se encuentran al límite del hiperplano de separación. En la figura 3.4 se puede ver en donde se encuentran los vectores de soporte señalizados con los símbolos rellenos.

2. **Bosques Aleatorios (Random Forests - RF):** RF es un algoritmo de clasificación que consiste en una colección de clasificadores estructurados como árboles, $h(x, \theta_k, k = 1, ...)$ donde los θ_k son vectores aleatorios idénticamente distribuidos y cada árbol emite un voto para la unidad más popular en la clase de entrada x [4].

   Para poder cumplir con la clasificación, el Algoritmo RF debe crear un conjunto de entrenamiento el cual se selecciona aleatoriamente con reemplazo, es decir, que no todos los datos del conjunto original estarán en el conjunto de entrenamiento, los datos que no fueron escogidos serán elegido para el conjunto de validación. En cada punto de división del árbol (nodos), la búsqueda de la mejor característica para dividir los datos no se realiza sobre todas las características sino sobre un subconjunto m. Se busca la mejor división de los datos de entrenamiento teniendo en cuenta sólo las m características que fueron elegidas aleatoriamente. Se repite varias veces los procesos anteriores, de forma que se tienen un conjunto de árboles de decisión entrenados sobre diferentes conjuntos de datos. Una vez que el algoritmo se encuentre entrenado, la clasificación es realizada por el voto mayoritario del conjunto de árboles.

3.2.4. Evaluación del rendimiento

La etapa de evaluación es un aspecto fundamental del aprendizaje para poder predecir el comportamiento de futuros objetos desconocidos. A través del análisis de tasas de acierto o errores se puede evaluar al clasificador en este sentido el objetivo de la evaluación es buscar un mínimo de errores y fallas.

3.2.4.1. Precisión

La precisión es el resultado de dividir el número de clasificaciones de muestras correctas por el número de muestras totales. En general la precisión es una buena estimación de como se comportara el modelo para datos desconocidos similares a los datos de prueba. En [32] , la precisión es calculada como:

$$precisión = \frac{número\ \ de\ \ muestras\ \ correctamente\ \ clasificadas}{número\ \ total\ \ de\ \ muestras}$$

Cuando se quiera medir la precisión se debe tener extremo cuidado al calcular la precisión sobre el mismo conjunto de datos utilizados para generar el modelo, ya que es probable obtener una precisión mayor a la real, es decir, serán estimaciones muy óptimas por haber utilizado los mismo ejemplos en el entrenamiento del algoritmo y en su comprobación. Por lo mencionado anteriormente se pretende que en la mayoría de los casos se debe seleccionar una porción de los datos para estimar el modelo y posteriormente comprobar su validez con el resto de los datos.

3.2.4.2. Técnicas de evaluación

Para evaluar un modelo se parten los datos en dos conjuntos. Por un lado, se tiene el conjunto de entrenamiento (training set), este conjunto servirá para enseñar al modelo cuál es el comportamiento del sistema, haciéndose una clasificación por el análisis de dichas instancias. Por otro lado, se tiene el conjunto de prueba (test set), que será el conjunto sobre el que se aplicarán los métodos una vez adquirido el conocimiento previo a través del conjunto de entrenamiento. A continuación se describen tres métodos fundamentales para la validación.

Validación Simple: Utiliza un conjunto de muestras para construir el modelo del clasificador, y otro diferente para estimar el error, con el fin de eliminar el efecto de la sobreestimación. De entre la variedad de porcentajes utilizados, uno de los más frecuentes es tomar 2/3 de las muestras para el proceso de aprendizaje y el 1/3 restante para comprobar el error del clasificador. El hecho de que sólo se utiliza una parte de las muestras disponibles para llevar a cabo el aprendizaje es el inconveniente principal de esta técnica, al considerar que se pierde información útil en el proceso de inducción del clasificador. En [41] se señala que la situación de evaluación se deteriora si el número de muestras para construir el modelo es muy reducido.

Validación Cruzada: Se utiliza para evitar la ocultación de parte de las muestras al algoritmo de inducción y la consiguiente perdida de información. En muchos casos se denomina **Validación cruzada con k pliegues** cuando el conjunto de datos se divide en k pliegues[10] mutuamente exclusivas, donde para cada pliegue existe un número similar de muestras. En cada evaluación, se deja uno de los subconjuntos para la prueba, y se entrena el sistema con los k-1 restantes. El número de pliegues más utilizado es el $k = 10$ en donde se estima que se logran mejores resultados [41]. Un caso particular de este método de evaluación es la **Validación cruzada dejando uno fuera** (leaving-one-out cross validation), donde k es igual al número de muestras del conjunto de datos. En este caso, el clasificador se entrena con todas las muestras menos una que se deja fuera para realizar la prueba. El mayor inconveniente de este método es el alto coste computacional que supone el aprendizaje del clasificador k veces, por lo que no se suele utilizarse cuando el número de muestras es elevado o el proceso de entrenamiento del clasificador es computacionalmente costoso.

Bootstrapping: Esta es una técnica de evaluación muy usada para conjuntos de datos pequeños o con la maldición de dimensionalidad (que es cuando el conjunto de datos tiene una gran diferencia entre el número de características frente al número de particiones) a diferencia de las técnicas de validación simple y validación cruzada que se muestran deficientes para la validación de pocos datos. Bootstrapping aumenta la eficiencia de validación equiparable a un aumento en el tamaño de las muestras en un 63,2 %. Si tenemos un conjunto de datos m, se realiza un muestreo aleatorio con reposición de m, para formar el conjunto de aprendizaje. El proceso se repite un número determinado de veces y luego se ejecuta como un caso de validación cruzada (promediando las precisiones).

En este capítulo se pretendió dar un aspecto global del procesado de los datos, y poder precisar en dónde se sitúa concretamente la selección de características, dentro del proceso de extracción del conocimiento para obtener la clasificación. En los siguientes capítulos se tratarán conceptos relacionados con la selección de características continuando con su aplicación en la Bioinformática exclusivamente en la Interacción Proteína-Proteína.

[10] Los pliegues son las particiones que se hacen sobre el conjunto de datos.

Capítulo 4

Selección de Características

La creciente cantidad de datos debido a los avances tecnológicos en los últimos años, ha impulsado el desarrollo de nuevas técnicas para adquirir conocimiento. Debido a esta gran cantidad de datos han surgido diversos estudios en donde se busca reducir esta dimensionalidad y sólo utilizar datos necesariamente validos que ayuden a comprender el comportamiento sin la necesidad de manipular el espacio completo de datos. Poseer una cantidad de datos elevado puede llevar a tener problemas en tiempos de ejecución muy elevados y que existan características que no aporten a la clasificación.

¿Porqué los algoritmos de selección de características surgen en el estudio del conocimiento?. Esto es una pregunta que surgió alrededor de la década de los 70, en donde variados investigadores empezaron a desarrollar estudios sobre algoritmos que permitieran obtener a través de un conjunto Y obtener un subconjunto X tal que se obtenga como resultado un subconjunto con características que no disminuyan la credibilidad del conjunto y elimine las características no relevantes y redundantes. Los algoritmos de selección de características se ven aplicados en distintas áreas del conocimiento, entre ellas se encuentran directamente relacionadas tales como el reconocimiento de patrones, aprendizaje automático y la minería de datos. En el transcurso del tiempo los algoritmos de selección de características han sido utilizados para diferentes propósitos como la clasificación de texto, recuperación de imágenes, dirección de relaciones con clientes, bioinformática, entre otros.

4.1. Conocimientos previos

El proceso de selección de características parte por escoger un subconjunto de características desde el conjunto original, obteniendo un subconjunto que sea relevante y que logre el máximo rendimiento con un esfuerzo menor que utilizar el conjunto completo reduciendo la complejidad del procesamiento computacional.

Las características irrelevantes y redundantes juegan un papel desfavorable en el proceso de clasificación, es por esto que los algoritmos de selección de características buscan reducir su importancia. El poseer más características, implica que se deben generar más instancias para garantizar la fiabilidad de los patrones por lo que las características irrelevantes y redundantes confunden al algoritmo de aprendizaje (un clasificador es menos exacto que otros que aprenden sólo de características relevantes). Además, el poseer más características de datos irrelevantes y redundantes suele llevar a que el clasificador se vuelva más complejo, lo que dificulta el entendimiento de los resultados.

La selección de características por lo mencionado anteriormente puede aportar en [41]:

1. Reducir la dimensión de datos, lo que implica que los algoritmos pueden aprender más rápido.
2. Mejorar la precisión, ya que el clasificador generaliza mejor.
3. Entrega resultados más simples y fáciles de entender.

4.2. Proceso General

La selección de características según [24], se puede representar como un problema de búsqueda en un espacio de estados, en donde cada estado corresponde a un subconjunto de características y el espacio considera todos los posibles subconjuntos que se pueden generar ver Figura 4.1. Por ejemplo en un conjunto de 4 características ($n = 4$), el espacio total se compone por 16 subconjuntos (2^n). En estricto rigor la selección de características se puede entender como el recorrido del espacio total hasta encontrar un subconjunto que optimice alguna función definida sobre un conjunto de características.

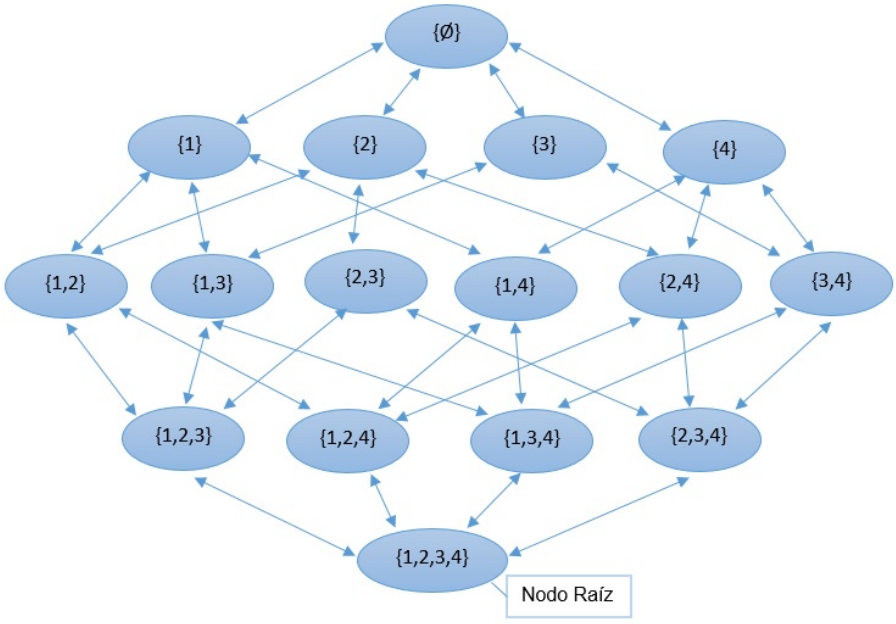


Figura 4.1: Espacio de búsqueda.

Como se puede ver en la Figura 4.1 el problema de selección de características se puede representar a través de una búsqueda en un grafo dirigido, donde el nodo raíz corresponde al conjunto de todas las características. El número total de posibles subconjuntos de un conjunto de características de n elementos es (2^n). En el grafo cada nodo corresponde a un subconjunto de características [45].

En general los procedimientos de selección de características se distinguen cuatro etapas (ver Figura 4.2):

1. Procedimiento de Selección: En esta etapa se determina el posible subconjunto de características para realizar la representación del problema [43]. El procedimiento de selección puede empezar con; (1) un conjunto vacío de características; (2)un conjunto con todas las características; (3) un subconjunto aleatorio de características. En los dos primeros casos, las características son iterativamente agregadas o removidas, mientras que en el último caso, las características se pueden añadir o remover iterativamente o producirse aleatoriamente [24].

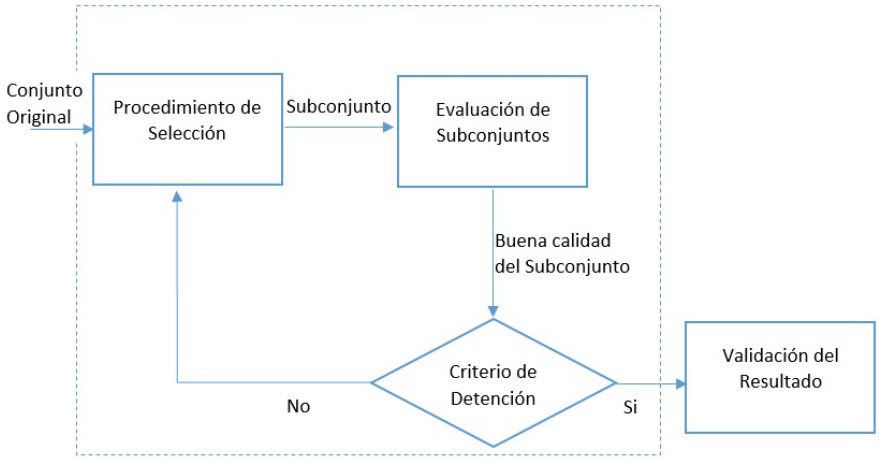


Figura 4.2: Proceso de selección de características [17; 43; 6].

2. Evaluación de Subconjunto: Mide la calidad de un subconjunto producido en la etapa anterior, y compara los valores obtenidos con el mejor anterior. Si se encuentra que es mejor se sustituye con el anterior subconjunto.
3. Criterio de Detención: Sin un criterio de termino adecuado el proceso de selección de características se puede ejecutar a través de todo el espacio de subconjuntos. Según la investigación de Dash-Liu [6] el criterio de termino se puede diferenciar en:
   a) Criterios basados en el Procedimiento de Selección:
      1) Selección de un número predefinido de características.
      2) Número predefinido de iteraciones.
   b) Criterios basados en la Función de Evaluación:
      1) Si al agregar o eliminar cualquier características no se produce un mejor subconjunto.
      2) Si un subconjunto óptimo es obtenido de acuerdo a la función de evaluación se detiene.
   c) Validación del Resultado: Esta etapa no es propiamente tal de la selección de características. Lo que se busca es comprobar la válidez de un subconjunto seleccionado mediante la realización de diferentes pruebas, comprobando los resultados con los

resultados previamente establecidos.

4.3. Subconjuntos

La selección de características cuenta con un punto de partida en el cual puede ser el conjunto completo de característica, el conjunto vacío o algún punto intermedio. Cuando se evalúa el primer subconjunto, los otros subconjuntos se irán examinando dependiendo de una dirección de búsqueda que puede ser hacia adelante, hacia atrás, aleatoria o alguna combinación de estas como la bidireccional. El proceso de búsqueda terminará cuando se recorra todo el espacio o cuando se cumpla una condición de termino.

4.3.1. Dirección de búsqueda

Como se mencionó anteriormente la dirección de búsqueda, guarda relación entre el subconjunto evaluado con el siguientem subconjunto, realizando un recorrido a través del espacio de subconjuntos. A continuación se describen las diferentes dirección de búsqueda:

Secuencial Hacia Adelante: En la dirección hacia adelante, la búsqueda comienza con un conjunto vacío de características, este conjunto se irá llenando a medida que se vaya eligiendo la mejor característica del conjunto original que no haya sido elegida anteriormente, esta elección se hace en base a un criterio de evaluación. El algoritmo con este tipo de búsqueda se ejecutará hasta cumplir con un criterio de termino, el cual puede ser un número predeterminado de características señaladas por el usuario, en el caso que no se señale ningún criterio de termino se ejecutará hasta conseguir un ranking de características.

Secuencial Hacia Atrás: En la dirección hacia atrás, la búsqueda comienza con el conjunto completo de características, en cada iteración se elige la característica menos relevante de entre las características originales en base al criterio de evaluación y se elimina. De la misma forma que la dirección de búsqueda hacia adelante, este algoritmo se ejecutará hasta cumplir con un criterio de termino, el cual puede ser un número predeterminado de características señaladas por el usuario, en el caso que no tener un

criterio de termino el algoritmo se ejecutará hasta al final obteniendo un ranking de características pero este será ordenando según su irrelevancia.

Aleatoria: Como su nombre lo dice la búsqueda aleatoria parte de un inicio aleatorio el cual no es definido y cambia en cada ejecución, por ende no se conoce su punto de partida ni cual será su siguiente subconjunto seleccionado para ser evaluado. Se pasa de un subconjunto a otro sin ningún orden establecido, añadiendo o eliminado uno o varias características. Lo que se pretende es evitar elegir subconjunto de características que sean óptimos locales, como sucede en los casos anteriores. En general el criterio de termino suele ser un número predefinido de iteraciones.

Otros esquemas se pueden obtener variando o mezclando algunos de los anteriores, como el Bidireccional, que realiza la búsqueda hacia adelante y hacia atrás al mismo tiempo, o el esquema Compuesto, que aplica una serie de pasos consecutivos hacia adelante y otra serie de pasos consecutivos hacia atrás [41].

4.3.2. Estrategia de búsqueda

Dado que un conjunto está determinado por n características el total de subconjuntos candidatos para obtener el óptimo es 2^n. Realizar una búsqueda completa en cualquier conjunto de características es totalmente ineficiente, incluso para conjuntos de tamaños medianos, siendo necesario aplicar estrategias de búsqueda para reducir el tamaño de búsqueda. Existen diferentes enfoques que serán explicados a continuación:

Completa: Esta búsqueda garantiza la localización de un resultado óptimo pero depende del criterio de evaluación utilizado. Si para seleccionar los subconjuntos óptimos se tienen que examinar todos los posibles subconjuntos es una búsqueda completamente exhaustiva. Sin embargo, Schlimmer [44] argumenta[11] que:

> Simplemente porque la búsqueda debe ser completa, no significa que tiene que ser exhaustiva.
>
> (Schlimmer, 1993)

[11] Just because the search must be complete does not mean that it must be exhaustive. (Schlimmer, 1993).

Entonces según el criterio de evaluación utilizado se puede lograr no recorrer todos subconjuntos del espacio posibles $O(2^n)$. Algunas implementaciones de búsqueda completa pero no exhaustiva son Ramificación y Poda (Branch & Bound - B&B).

Heurística: Son estrategias que realizan una búsqueda parcial a través del espacio de características, es decir, no visitan todos los subconjunto del espacio. Este tipo de algoritmo presenta una rapidez mayor en su ejecución en comparación a los de búsqueda completa ya que su dimensión de búsqueda es menor. En este tipo de búsqueda se pueden encontrar algoritmos con dirección de búsqueda hacia atrás, hacia adelante o bidireccionales. En [6] señala que el orden del espacio de búsqueda es $O(2^n)$ o menor.

Aleatoria: Las dos estrategias de búsqueda anteriormente mencionada son deterministas, es decir, no importa las veces que se ejecuten devuelven el mismo resultado. Las búsquedas aleatorias en cambio son estocásticas, es decir, en diferentes ejecuciones se pueden obtener resultados diferentes porque están sometidas al azar. Las búsquedas aleatorias busca a través del espacio formado por las características de manera aleatoria, es decir, no existe un estado siguiente o anterior que se pueda determinar según alguna regla. Con esta búsqueda se quiere llegar a diferentes puntos del espacio global de características independientemente de que sean subconjuntos peores a los ya seleccionados, con la idea de abarcar más localizaciones en el espacio. En este grupo podemos encontrar algoritmos tales como Algoritmos Genéticos y Redes Neuronales para selección de características.

4.3.3. Funciones de evaluación de características

Un subconjunto óptimo es siempre relativo a un cierto criterio de evaluación. En Dash y Liu [6] se consideran cinco categorías: distancia, información, dependencia, consistencia y Tasa de error del clasificador. En la siguiente sección se dará una breve descripción de cada una.

Medida de Distancia: Conocidas también como medidas de separabilidad, divergencia o discriminación. Estas medidas estiman la capacidad de un subconjunto de caracte-

rísticas en separar las clases. Para un problema de dos clases, una característica X se prefiere de otro Y, siempre que X tenga una mayor diferencia entre las probabilidades entre las clases que Y. Si X e Y la diferencia es cero son indistinguibles. En [41] se nombran algunos ejemplos de medidas de distancia son: Euclidea, Manhattan, Mahalanobis, Bhattaacharya, Kullback-Liebler, Kolmogorov, Chernoff, etc.

Medidas de Información: Esta medida típicamente determina la ganancia de información a partir de una característica. La ganancia de información de una característica X se define como la diferencia entre la incertidumbre anterior y la esperada de X. Si la ganancia de informacion de X es mayor que la de Y se prefiere X. En [41] se señala que entre las medidas de información más frecuentes se encuentran; la entropía de Shannon, de Renyi, de grado α, cuadrática, estrictamente cóncava y de Daroczy, MDLC e información mutua.

Medidas de Dependencia: Conocidas también como medidas de correlación o medidas de similitud. Miden la capacidad de predecir el valor de una variable a partir del valor de otra. En la selección de características para la clasificación, se busca con qué medida de correlación está la característica asociada una clase. Una característica X se prefiere de otra Y si la asociación entre la característica X y la clase C es mayor que la asociación entre Y y C. La evaluación de característica mediante medidas de dependencia, está muy relacionada con la evaluación según las medidas de información y distancia.

Medidas de Consistencia: Estas medidas se caracterizan por su fuerte dependencia con el conjunto de entrenamiento. Estas medidas tratan de encontrar un número mínimo de características que separan las clases de la manera más coherente desde el conjunto completo de características la cual satisfaga una tasa mínima de inconsistencia aceptable. El problema con esta medida es cuando las bases de datos se encuentran individualmente identificadas cada instancia con un identificador tal como el Rut, patente, etc. lo que produce que no exista inconsistencia en los datos. Una solución a lo anterior es dejar esta característica fuera del proceso de selección si está identificado.

Medidas de Tasa de error: En el aprendizaje supervisado, el objetivo principal de un clasificador es maximizar la precisión en la predicción de nuevas muestras, esto hace que la precisión sea aceptada y muy utilizada como medida de evaluación. Como se seleccionan las características mediante la utilización del clasificador que posteriormente se emplea en la predicción de las etiquetas de la clase para los ejemplos desconocidos, el nivel de precisión es alto, pero se añade un elevado coste computacional al algoritmo de selección. En [6], se dice que los métodos que utilizan este tipo de función de evaluación son llamados "métodos de contenedor", es decir, el clasificador es la función de evaluación.

Función de Evaluación	Generalidad	Tiempo de complejidad de reconocimiento	Precisión
Medida de distancia	Si	Bajo	*
Medida de Información	Si	Bajo	*
Medida de Dependencia	Si	Bajo	*
Medida de Consistencia	Si	Moderado	*
Medidas de Tasa de error	No	Alto	Muy Alta

Tabla 4.1: Comparación Tipos de criterios [6].

La tabla 4.1 muestra una comparación de varias funciones de evaluación, independiente del tipo de Procedimiento de Selección utilizado. Los parámetros utilizados para la comparación son:

a) Generalidad: es el subconjunto seleccionado por diferentes clasificadores.

b) Tiempo de complejidad: es el tiempo transcurrido para seleccionar el subconjunto de características.

c) Precisión: que tan buena es la predicción usando el subconjunto seleccionado.

En la ultima columna el símbolo * indica que no se puede determinar una precisión sobre la función de evaluación ya que dependen del conjunto de datos y clasificador utilizado.

4.4. Algoritmos de Selección

Hasta ahora se ha visto para qué sirve los algoritmos de selección de características y cuál es su uso en diversos estudios de la extracción del conocimiento. La cantidad de algoritmos de selección existentes es alta, en [41] se dice que existen sobre 75 algoritmos de selección, además hay una gran variedad de algoritmos de selección que son modificaciones de los algoritmos originales. En la Figura 4.3 se muestra un resumen de los métodos de selección de característica basado en la estrategia de búsqueda presentada en la sección 4.3.2.

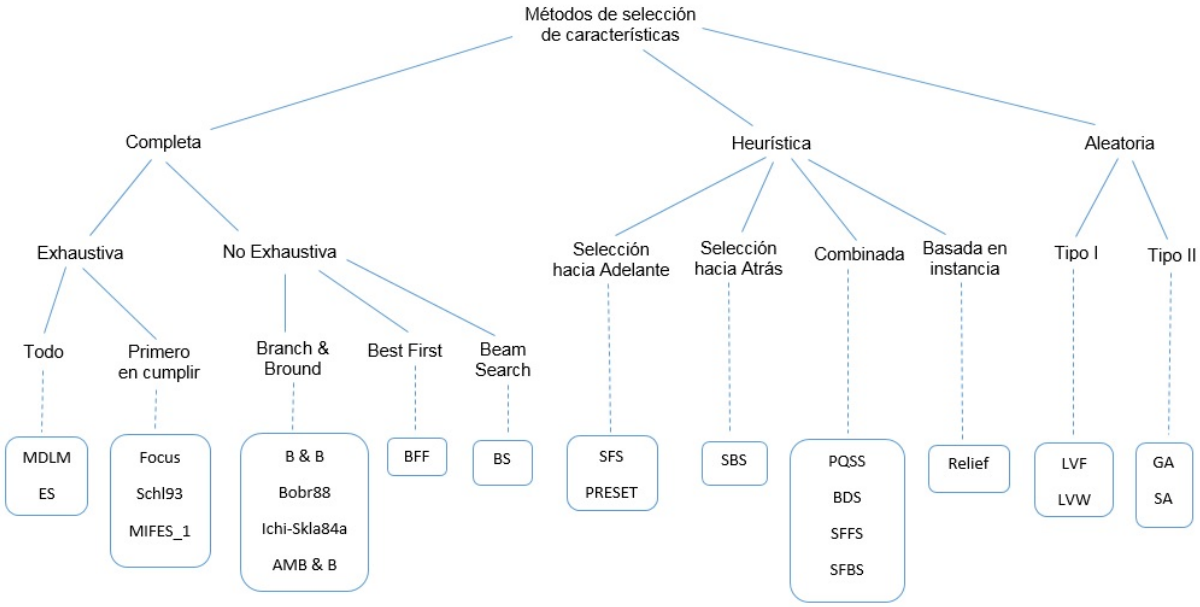


Figura 4.3: Resumen de los métodos de selección de características. Adaptada de [6].

En la Figura 4.3 se puede apreciar que los procedimientos completos se subdividen en exhaustiva y no exhaustiva; en la categoría exhaustiva un método puede evaluar todos los subconjuntos hasta encontrar el óptimo como es el caso de Búsqueda Exhaustiva (Exhaustive Search - ES), o en el otro caso hacer una búsqueda hasta que un subconjunto óptimo se encuentre como es el caso del algoritmo Focus. En la categoría no exhaustiva encontramos algoritmos tales como branch & bound, Best First y Beam Search. Los procedimientos heurísticos se subdividen en selección hacia adelante, selección hacia atrás, combinada y

basado en instancias. De igual forma, los procedimientos aleatorios se agrupan en tipo I y tipo II; en tipo I la probabilidad de un subconjunto que se genera se mantiene constante y es la misma para todos los subconjuntos; en tipo II la probabilidad cambia a medida que se ejecuta el programa.

Como se ha demostrado anteriormente existe un gran número de algoritmos de selección de características que han sido utilizados en diferentes investigaciones, pero aún no hay un estándar que defina que algoritmo escoger en base a lo que se necesita investigar. Un estudio realizado por Kudo y Sklansky [23] pretende recomendar un algoritmo de selección de características en base a la cantidad de datos del conjunto (ver Apéndice B).

Como se puede apreciar en la Figura 4.3 y la Figura 4.4, hay diferentes formas de categorizar y agrupar a los algoritmos de selección de características, ambas son correctas pero miradas desde diferente enfoques.

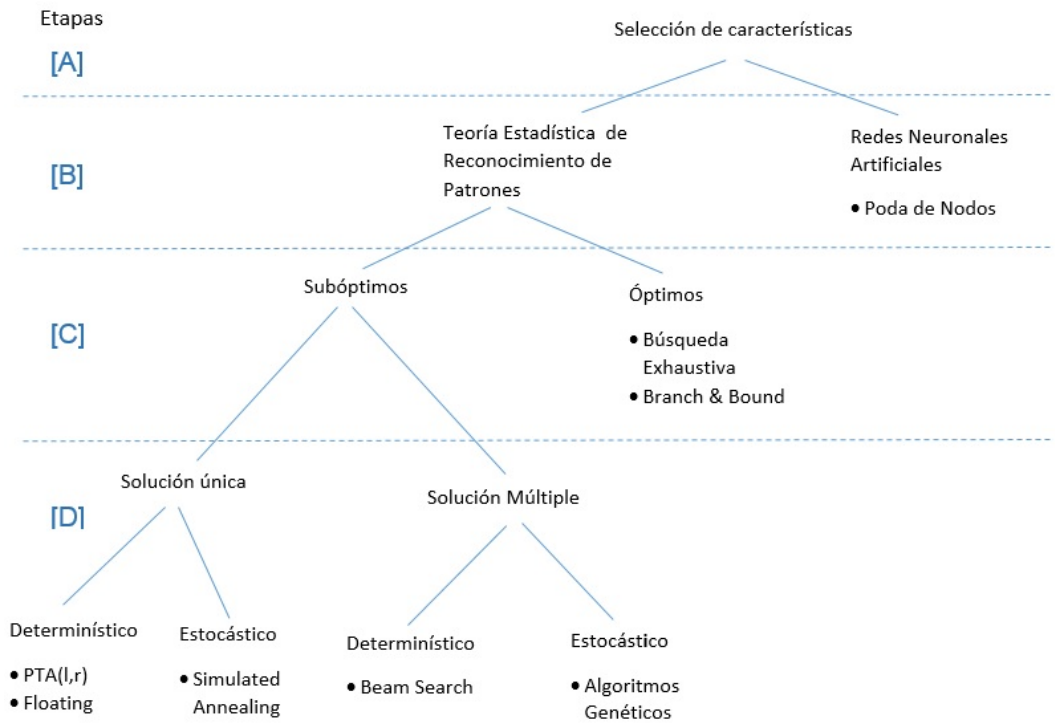


Figura 4.4: Taxonomía algoritmo de selección de características. Adaptada de [20].

En la Figura 4.4 se muestra una taxonomía propuesta por Jain y Zongker [20] sobre los métodos de selección de algoritmos. En la primer etapa [A], la selección de característica se divide en dos métodos los cuales conforman la etapa [B] los basados en la teoría estadística de reconocimientos de patrones y aquellos basados en redes neuronales artificiales. En la etapa [C], lo métodos basados en la teoría estadística de reconocimientos de patrones se dividen en los que obtienen una solución óptima y los que obtienen soluciones suboptimas. Además los métodos suboptimos se dividen para formar la etapa [D], en donde se diferencian los algoritmos que sólo trabajan sobre un subconjunto y realizan modificaciones sobre él (solución única), al contrario de los que crean una población de subconjuntos (solución múltiple). La última distinción que se puede hacer es sobre los métodos suboptimos en donde el algoritmo puede ser determinista, es decir, produce el mismo subconjunto de un problema cada vez que se ejecuta y los estocásticos que tienen un comportamiento aleatorio que podrían producir diferentes subconjuntos cada vez que se ejecutan.
Como se aprecia en la taxonomía y en presencia del estudio de esta investigación sobre algoritmos de selección se destacará algunos algoritmos que suelen ser nombrados en la literatura. A partir de las Figuras 4.3 y 4.4 se implementó una comparación entre los algoritmos de selección en la cual se señala que método utiliza y que solución de salida entrega cada algoritmo (ver Tabla 4.2).

Algoritmo de selección de características	Método	Solución
Ramificación y Poda	Completo no exhaustivo	Óptimo
Búsqueda Secuencial Hacia Adelante	Heurístico	Subóptimo
Algoritmo Genético	Aleatorio	Subóptimo

Tabla 4.2: Comparación algoritmos.

Para la definición de los algoritmos se usarán ilustraciones tomadas del trabajo de Doak [8] en donde se puede apreciar de forma gráfica y descriptiva el comportamiento de los algoritmos en base al conjunto de características.

4.4.1. Ramificación y Poda (Branch and Bound - B&B)

El Algoritmo Ramificación y Poda (B&B) según la Tabla 4.2 se caracteriza por pertenecer a los completos no exhaustivos y entregar como salida un subconjunto óptimo. El algoritmo B&B garantiza encontrar un conjunto de características óptimo siempre y cuando el criterio de evaluación definido como J sea monótono. La monotonicidad aplicada al criterio de evaluación en este caso quiere decir que si se elimina cualquier característica o subconjunto desde el conjunto global, todos los nodos sucesivos a él tendrán valores menores y no pueden ser soluciones óptimas. En B&B se busca que el subconjunto óptimo sea lo más pequeño posible y disponer de un umbral que permita discriminar con el valor calculado por el criterio de evaluación, por ejemplo, si la tasa de error se ocupa como criterio de evaluación, los subconjuntos que tengan una tasa de error mayor que el umbral serán desechados y no se consideran para el subconjunto óptimo (Poda). El algoritmo B&B por lo general se interpreta como un árbol en donde cada rama representa una posible solución. El algoritmo comienza su búsqueda desde el conjunto completo de características. Continúa su búsqueda por las ramificaciones que se van creando y elimina las características que no cumplan con el umbral o se alejen del óptimo, es decir, que poda la rama del árbol para no continuar por ese camino desperdiciando recursos y procesos. La propiedad de monotonicidad se aplica en este caso, ya que elimina cualquier característica del conjunto que se sabe que no va a mejorar su valor y por consiguiente el espacio de búsqueda se ve reducido. Entonces se debe volver a un estado anterior en donde no se viole el umbral para lograr el óptimo y buscar otro camino. Dicho lo anterior, se presume que el umbral aplicado al criterio de evaluación tiene una gran relevancia dentro del algoritmo B&B, ya que permite ignorar durante la búsqueda los subconjuntos que no cumplan con el criterio de evaluación, esto hace que el algoritmo B&B se vuelve dependiente del umbral para obtener un buen resultado. Otro problema es que el algoritmo realiza una búsqueda sobre la región que este resultando factible, la que crece en función de 2^{n-m} donde n es la dimensión del espacio original y m es la dimensión de entradas seleccionas, por lo que aunque se reduce el espacio total la complejidad sigue siendo exponencial.

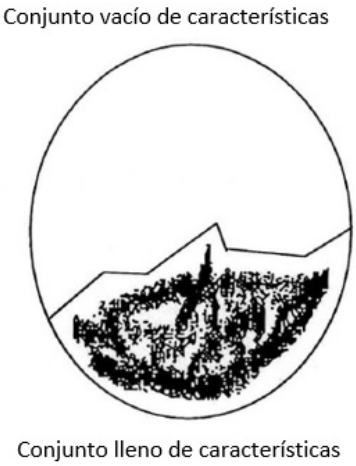


Figura 4.5: Branch and Bound [8].

4.4.2. Búsqueda Secuencial Hacia Adelante (Sequential Forward Search - SFS)

El Algoritmo Búsqueda Secuencial Hacia Adelante (SFS) se ve en la Tabla 4.2 como un método heurístico y se caracteriza por entregar una salida sub-óptima de una solución. Este algoritmo consta de tres elementos esenciales el conjunto de característica Y, el conjunto vacío X en donde se irán agregando las características mejor evaluadas y el criterio de evaluación J. El algoritmo comienza desde un conjunto vacío en el cual se irán agregando características que obtengan el mejor resultado dependiendo de un criterio de evaluación $J(X)$. Las características que aún no se han agregado son evaluados una por una en combinación con las ya seleccionadas formando nuevos subconjuntos para ser evaluados por el criterio de evaluación $J(X)$, el subconjunto con la mejor evaluación se agrega al conjunto que va almacenando las características con mejores resultados. Este proceso se repite hasta cumplir con el criterio de termino o en el caso de no poseer algún criterio de termino se ejecuta con todas las características logrando un ranking de características ordenadas por orden de evaluación. Este algoritmo presenta limitaciones al momento de eliminar características ya seleccionadas para poder volver a formar un subconjunto mejor al ya creado. Esto se traduce en que el algoritmo no revisa todos los estados posibles, causado por la

incapacidad de dar marcha atrás.

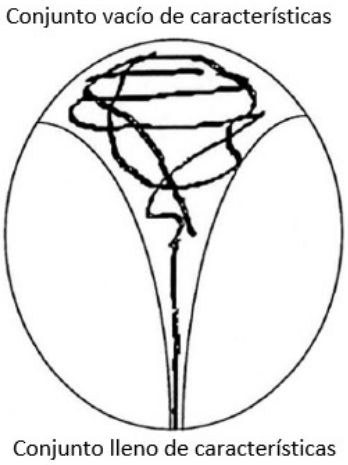


Figura 4.6: Sequential Forward Search [8].

4.4.3. Algoritmo Genético (Genetic algorithm - GA)

Los Algoritmos Genéticos (GA) según la Tabla 4.2 se caracterizan por pertenecer a los métodos aleatorios y se caracterizan por entregar como salida un método subóptimo de muchas soluciones. Los algoritmos genéticos están inspirados en el concepto de evolución biológica. La idea principal de este algoritmo es que cada individuo de una población represente una posible solución al problema que se desea solucionar y la adaptación que tenga a la evolución de la población. Con lo anterior se irán generando nuevas soluciones dado las combinaciones que se vayan generando entre poblaciones. La esencia general de un algoritmo genético es comenzar con la construcción de una posible población inicial que pueda representar una solución al problema; luego el algoritmo comienza a iterar, las iteraciones son conocidas como generaciones, durante las cuales se crean y evalúan nuevas poblaciones. Cada población crea un conjunto de apareamiento, en donde usualmente se encuentran los mejores candidatos de la población; luego se crean descendencia o una nueva generación este proceso es a través de los conjuntos de apareamiento, proceso conocido como cruzamiento, en donde la combinación de dos individuos que pertenecen a diferentes

poblaciones se realiza. La descendencia es creada por algún criterio aleatorio en donde se eligen padres al azar para crear la nueva generación, por lo general, se hace la selección aleatoria para que se permita crear nuevas generaciones a partir de diferentes padres y disminuya la tasa de repetición. Finalmente las poblaciones son creadas a partir de los mejores individuos de las poblaciones o descendencia.

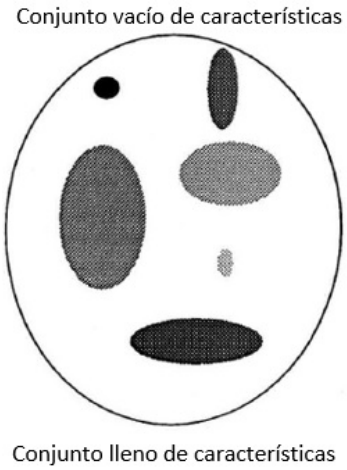


Figura 4.7: Algoritmo Genético [8].

Capítulo 5

Estudio del problema

5.1. Hitos y estudios

Las Interacciones Proteína-Proteína (IPP) son muy importantes en el ámbito biológico y de la vida, existen muchas investigaciones que buscan predecir la interacción entre proteínas porque en ellas radica las funciones que cumplen. La ciencia de la computación en el último tiempo a sido muy utilizada para poder ayudar a comprender las IPP y su zona de interacción, es por esto que muchas investigaciones comenzaron a implementar técnicas de reconocimiento de patrones en las bases de datos como en [18], los cuales utilizan Máquinas de soporte de vectores (Support Vector Machine - SVM) para clasificar la interacción entre complejos, en [32] se propone un nuevo método basado en los perfiles filogenéticos[12] para poder predecir la interacción de proteínas en base a la información de la secuencia primaria de una proteína, para este estudio se utilizaron diferentes métodos de clasificación sobre los conjuntos de datos de perfiles filogenéticos con el objetivo de encontrar un patrón de interacción, se obtuvo la mayor precisión en la clasificación con Bosques Aleatorios (Random Forest - RF) con un 76.93 %. La mayoría de las investigaciones se diferencian en que escogen diferentes características para clasificar las interacciones entre proteínas. La investigación desarrollada por Gutiérrez-Bunster [14] utiliza las propiedades de caracterís-

[12] Estudiando el perfil filogenético de dos proteínas se puede obtener información sobre el grado de similitud de su historia evolutiva. Cuanto más similar sea esta, más posibilidades existen de que esas dos proteínas interaccionen.

ticas energéticas de la superficie de interacción con la cual obtiene un 81 % de precisión, siguiendo con este trabajo en Gutiérrez-Bunster et al. [15] obtiene un 86.6 %, ésta mejoría se atribuye al aumento de características energéticas para la clasificación de los complejos. Para este capítulo es esencial presentar los hitos más importantes que fueron desarrollados años atrás pero siguen siendo de base para muchas investigaciones, el primer hito se atribuye a Mintseris y Weng [30] con su investigación se obtuvo los complejos separados por tiempo de duración de la interacción entre proteínas, el segundo hito se atribuye a Protein Data Bank (PDB) [2] con el cual se pudo obtener los complejos en su forma tridimensional para poder pasar al paso final que es a través de la aplicación FastContact [5] la cual entrega las propiedades energéticas que ocurren en la superficie de interacción entre proteínas. Como se mencionó anteriormente, en los estudios realizados por Gutiérrez-Bunster [14; 15] se logró conseguir una alta precisión en la clasificación de las IPP, utilizando estos mismos hitos como bases de conocimiento. Cada uno de los hitos mencionados serán presentados en las siguientes secciones, en las cuales se centrará en explicar su función y utilización.

5.2. Metodología propuesta

Han surgido variadas investigaciones sobre la superficie de interacción entre las proteínas pero la presente investigación es impulsada por las investigaciones realizadas por Gutiérrez-Bunster en [14] y en [15]. En esta sección se presentará la metodología llevada a cabo para la obtención de los datos que serán utilizados para su posterior selección y clasificación.

5.3. Preparación de los Datos

En la preparación de los datos se tienen que obtener características consistentes de fuentes fiables para el desarrollo de la investigación, para esto se utilizo el trabajo realizado por Mintseris y Weng [30] en el cual se logró clasificar las proteínas por su tiempo de duración en la interacción y posteriormente se permitió generar un listado con estos complejos [31] el cual será usado para poder obtener la estructura tridimensional de las proteínas desde

la base de datos Protein Data Bank (PDB) [2] para finalmente conseguir las características energéticas que suceden en la superficie de interacción entre el ligando y el receptor (ver Figura 5.1).

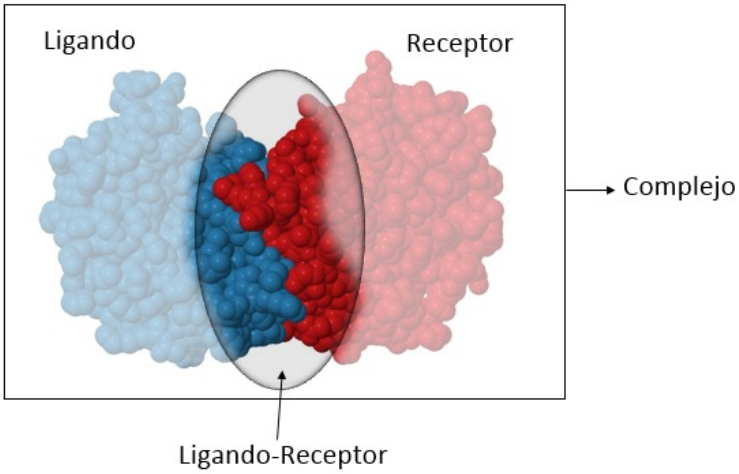


Figura 5.1: Complejo Proteico, identificación de la zona estudiada. PDB ID: 1TIM.

5.3.1. Obtención de Complejos

Desde el trabajo desarrollado por Mintseris y Weng [30] en donde se clasifican complejos de proteínas entre IPP permanentes y IPP transitorios (estos tipos de interacciones fueron definidos en la sección 2.2 en el trabajo realizado Nooren y Thornton [33]) se obtiene un listado de complejos que puede ser encontrado en [31].

El trabajo de Mintseris y Weng consta de estudiar la superficie de interacción entre proteínas, para esto ocuparon 326 complejos proteicos y se clasificaron de forma manual a través de laboratorio obteniendo como resultado un listado de 211 complejos de interacción transitoria y 115 complejos de interacción permanente.

5.3.2. Estructura tridimensional

Para obtener información estructural de las proteínas se debe acceder a las bases de datos de proteínas, de las cuales existe una gran variedad dependiendo de la información que se quiera extraer, en el trabajo de [42] se muestran diferentes bases de datos disponibles, diferenciándose por los datos que almacenan. En [14] se decidió utilizar la base de datos que ofreciera mayor accesibilidad y flexibilidad para el trabajo por lo que se utilizo la base de datos Protein Data Bank (PDB). El PDB es un repositorio mundial de datos estructurales sobre moléculas biológicas, mayoritariamente proteínas, el cual es sustentado por diferentes organizaciones que son responsable del depósito, mantenimiento, procesamiento y libre suministro de los datos biológicos para los investigadores. El PDB obtiene la estructura de las proteínas de forma experimental en laboratorio por lo que se utilizan diferentes métodos (vistos en la sección 2.2.1), tales como Cristalografía de rayos X con el cual han obtenido 107.790 estructuras, RMN con el cual han obtenido 11.469 estructuras, entre otros. Actualmente desde [2] se puede obtener que el PDB posee sobre las 120.000 estructuras de proteínas descubiertas.

El PDB almacena información estructural de las Proteínas que incluye las coordenadas en 3-D de los átomos de la molécula(s) que la componen. Estas coordenadas son la estructura terciaria de una proteína, la cual se encuentra relacionada en la función que cumple la proteína, por lo que al conocer esta estructura puede ayudar a entender la función de la proteína.

Para poder manipular las estructuras de proteínas desde el PDB se debe tener conocimientos sobre el formato de un archivo ".pdb", el cual se puede separa en dos secciones en la primera se puede apreciar la descripción inicial en donde se señala identificación, el autor, el método utilizado entre otros, mientras que en la segunda es la sección de átomos en donde se señala el átomo, coordenadas entre otros (ver Figura 5.2).

(a) Descripción Inicial

```
HEADER    HYDROLASE                                07-SEP-00   1FS0
TITLE     COMPLEX OF GAMMA/EPSILON ATP SYNTHASE FROM E.COLI
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: ATP SYNTHASE EPSILON SUBUNIT;
COMPND   3 CHAIN: E;
COMPND   4 EC: 3.6.1.34;
COMPND   5 MOL_ID: 2;
COMPND   6 MOLECULE: ATP SYNTHASE GAMMA SUBUNIT;
COMPND   7 CHAIN: G;
COMPND   8 EC: 3.6.1.34
SOURCE    MOL_ID: 1;
SOURCE   2 ORGANISM_SCIENTIFIC: ESCHERICHIA COLI;
SOURCE   3 ORGANISM_COMMON: BACTERIA;
SOURCE   4 MOL_ID: 2;
SOURCE   5 ORGANISM_SCIENTIFIC: ESCHERICHIA COLI;
SOURCE   6 ORGANISM_COMMON: BACTERIA
KEYWDS    ATP SYNTHASE, COILED COIL, GAMMA, EPSILON
EXPDTA    X-RAY DIFFRACTION
AUTHOR    M.C.J.WILCE,A.J.W.RODGERS
REVDAT   1   01-MAY-01 1FS0    0
JRNL        AUTH   M.C.J.WILCE,A.J.W.RODGERS
JRNL        TITL   STRUCTURE OF THE G-E COMPLEX OF ATP SYNTHASE
JRNL        REF    NAT.STRUCT.BIOL.                  V.   7  1051 2000
JRNL        REFN   ASTM NSBIEW  US ISSN 1072-8368
```

(b) Sección de átomos

```
ATOM      1  N   ALA E   1      23.208  25.479  38.434  1.00 69.40           N
ATOM      2  CA  ALA E   1      21.874  25.448  37.769  1.00 67.24           C
ATOM      3  C   ALA E   1      20.763  25.477  38.815  1.00 65.79           C
ATOM      4  O   ALA E   1      20.973  25.901  39.956  1.00 69.75           O
ATOM      5  CB  ALA E   1      21.747  24.196  36.899  1.00 48.49           C
ATOM      6  N   MET E   2      19.582  25.025  38.413  1.00 60.09           N
ATOM      7  CA  MET E   2      18.423  24.980  39.293  1.00 55.22           C
ATOM      8  C   MET E   2      17.880  26.343  39.721  1.00 46.03           C
ATOM      9  O   MET E   2      16.735  26.637  39.421  1.00 43.58           O
ATOM     10  CB  MET E   2      18.721  24.131  40.519  1.00 54.54           C
ATOM     11  CG  MET E   2      17.879  22.881  40.583  1.00 58.06           C
ATOM     12  SD  MET E   2      16.182  23.244  41.023  1.00 80.13           S
ATOM     13  CE  MET E   2      15.297  22.720  39.523  1.00 72.12           C
ATOM     14  N   THR E   3      18.663  27.168  40.414  1.00 40.24           N
ATOM     15  CA  THR E   3      18.152  28.483  40.811  1.00 41.75           C
ATOM     16  C   THR E   3      19.150  29.619  40.670  1.00 38.78           C
ATOM     17  O   THR E   3      20.353  29.395  40.566  1.00 41.40           O
ATOM     18  CB  THR E   3      17.698  28.522  42.290  1.00 31.30           C
ATOM     19  OG1 THR E   3      18.835  28.317  43.127  1.00 38.11           O
ATOM     20  CG2 THR E   3      16.632  27.458  42.583  1.00 32.78           C
ATOM     21  N   TYR E   4      18.622  30.844  40.656  1.00 35.40           N
ATOM     22  CA  TYR E   4      19.440  32.060  40.615  1.00 31.36           C
ATOM     23  C   TYR E   4      18.736  33.094  41.502  1.00 33.38           C
```

Figura 5.2: Formato Protein Data Bank (PDB). Obtenido usando ID: 1FS0.

La parte de interés de este archivo formato ".pdb" es la sección de átomos Figura 5.2(b) en donde se puede encontrar la información estructural del complejo, (para más información visitar Apéndice C).

Ahora que se tienen identificados los complejos que producen interacciones permanentes y transitorios obtenidos desde la lista de los complejos desde la investigación de Mintseris y Weng, se deben obtener las proteínas en un formato ".pdb". Este trabajo de obtención de la estructura de las proteínas fueron preparados manualmente en el trabajo de Gutiérrez-Bunster [14] para eliminar la duplicación de residuos. Finalmente se separa los complejos en dos diferentes archivos los cuales serán requeridos más adelante en la investigación para obtener las características energéticas.

5.3.3. Características energéticas

En la investigación de Gutiérrez-Bunster [14] como se ha mencionado anteriormente se centra en estudiar las propiedades energéticas que ocurren en la superficie de interacción, para poder clasificar entre IPP permanente e IPP transitoria.
Para obtener las características energéticas que ocurren en la superficie de interacción se utilizó una aplicación de análisis de Interacciones Proteína-Proteína (FastContact) desarrollada por Carlos Camacho y su Laboratorio [5]. FastContact entrega información sobre la contribución de los aminoácidos en la energía electrostática, energía de desolvatación y la energía libre de unión.

a) energía electrostática,
b) energía de desolvatación y
c) energía libre de unión.

Los datos de salida de FastContact entrega un archivo que proporciona diferentes energías por residuo separadas en siete partes cada parte entrega los 20 valores que más contribuyen y los 20 que menos contribuyen.
En la Tabla 5.1 se puede ver de forma descriptiva las partes que forman el archivo de salida del FastContact. En la primera Columna se dimensionan las siete partes entregadas por FastContact, la segunda columna se presentan los valores que entrega FastContact por cada residuo, siendo E el valor de la energía, R el valor del residuo y AA corresponde al aminoácido. Como se puede apreciar las últimas dos filas de la tabla se entregan cinco datos

ya que estás corresponden a la interacción entre el ligando y el receptor, perteneciendo los primeros R y AA al Ligando y los últimos al Receptor. La última columna muestra los valores que son utilizados siendo estos los valores de E y R porque presentan ser de tipo numérico y se descarta la utilización de los datos AA porque son de tipo carácter.

(Partes) Tipo de Contribución	Valores	Valores Usados
(1) Residuos contribuyen a la energía libre de unión	E, R, AA	E, R
(2) Residuos Ligando que contribuye a la energía de desolvatación	E, R, AA	E, R
(3) Residuos Ligando que contribuye a la energía electrostática	E, R, AA	E, R
(4) Residuos Receptor que contribuye a la energía de desolvatación	E, R, AA	E, R
(5) Residuos Receptor que contribuye a la energía electrostática	E, R, AA	E, R
(6) Residuos Receptor-Ligando contacto electrostático	E, R, AA, R, AA	E, R, R
(7) Residuos Receptor-Ligando contacto de energía libre	E, R, AA, R, AA	E, R, R

Tabla 5.1: Estructura de salida FastContact para cada complejo [15].

Las energías que se producen en la superficie de interacción pueden contribuir de manera repulsiva o atractiva entre las proteínas. Como se pudo observar anteriormente las energías entregadas por FastContact están asociadas a un residuo y a un aminoácido y en el caso de contacto (parte 6 y 7 de la salida FastContact) a dos residuos y dos aminoácidos, estos valores de la energía pueden ser negativos, es decir, contribuye a la interacción (atracción entre proteínas) siendo estas las energías más importantes para que se produzca la interacción, cuando el valor de la energía es cero no existe repulsión ni atracción y por último si las energías presentan valores positivos no contribuye a la interacción (repulsión entre proteínas).

5.3.4. Creación del conjunto de características

La información entregada por la aplicación FastContact permite construir el espacio de características energéticas con el cual se desea trabajar. En la sección anterior se explicó que esta información fue procesada y que sólo se ocuparán los datos que presenten valores numéricos. La información entregada de cada complejo se puede representar como un vector estático en donde X es el vector y X^t representa al vector transpuesto (ver Figura 5.3).

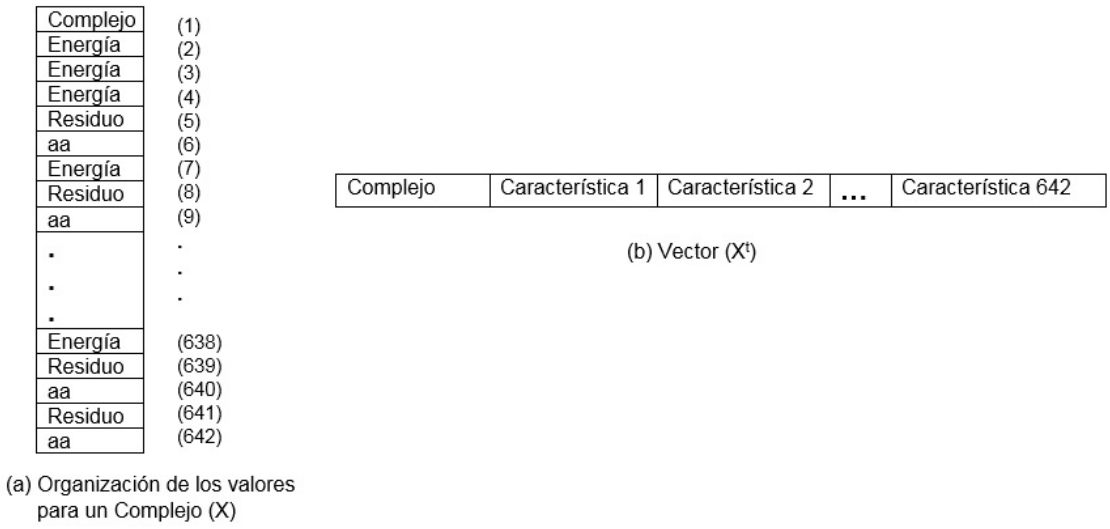


Figura 5.3: Representación de las características usadas desde FastContact [15].

Desde la Figura 5.3(b) se observa que el vector X^t se ve representado como $X^t = [x_1, x_2, ..., x_n]$ donde n es la cantidad de características y cada $x_1, x_2, ..., x_n$ representa a una característica que pertenece a un complejo C de clase $(c_1, c_2, ..., c_z)$ donde z es la cantidad de tipos de clases.

Para el procesamiento de las características obtenidas desde FastContact se utilizó una representación a través de una matriz M en la cual cada complejo ocupa una fila de la matriz, las características de energía (E) y/o residuo (R) ocupan las columnas de la matriz y además se debe considerar la columna inicial, la cual representa la clase a la que pertenece la característica la cual se diferencia como IPP permanente y IPP transitorio, ésto se puede

ver mas claro en la Ecuación 5.1, donde c representa a la clase y $x_{m,n}$ a las características.

$$M = \begin{bmatrix} c_1 & x_{1,1} & x_{1,2} & ... & x_{1,n} \\ c_2 & x_{2,1} & x_{2,2} & ... & x_{2,n} \\ ... & ... & ... & ... & ... \\ c_z & x_{m,1} & x_{m,2} & ... & x_{m,n} \end{bmatrix} \tag{5.1}$$

A continuación se explicará como se desarrollaron, las dos matrices en [14] y [15] que sustentan esta investigación de lo cual se puede adelantar su diferencia en la cantidad de características que poseen.

5.3.4.1. Creación de la primera Matriz

La primera Matriz desarrollada en [14] fue creada a partir de la salida estándar del Fast-Contact sin modificar, es decir, los 20 valores que más contribuyen y los 20 valores que menos contribuyen. La matriz tiene un total de 642 características, más la clase de interacción (permanente o transitorio) para cada uno de los 296 complejos de los que se diferencia 93 complejos permanentes y 203 transitorios. Finalmente la Matriz M_1 queda representada con una dimensión de 296x642 en la Ecuación 5.2.

$$M_1 = \begin{bmatrix} c_1 & x_{1,1} & x_{1,2} & ... & x_{1,642} \\ ... & ... & ... & ... & ... \\ c_1 & x_{93,1} & x_{93,2} & ... & x_{93,642} \\ c_2 & x_{94,1} & x_{94,2} & ... & x_{94,642} \\ ... & ... & ... & ... & ... \\ c_2 & x_{296,1} & x_{296,2} & ... & x_{296,642} \end{bmatrix} \tag{5.2}$$

En la Tabla 5.2 se observa con más detalle los valores utilizados en la creación de la Matriz M_1 en donde cada tipo de energía tiene los 20 residuos y los valores de las energías que más contribuyen en la interacción; y los 20 residuos y los valores de las energías que menos contribuyen, por lo tanto, se obtiene un total de 40 valores por cada tipo de energía. En

los primeros cinco tipos de energía se aprecian 2 características (E y R) obteniendo como total 80 características por cada uno y en los últimos dos tipos de energía se utilizan 3 características (E ,R Y R) obteniendo 120 características por cada uno.

			(20+, 20- Características)	
Cadena	Tipo de energía	Energía,residuos	Matrix 1	Total
Complejo	Energía electroestática			1
	Energía desolvatación			1
	Energía libre de unión	2(E y R)	40(20+,20-)	80
Ligando	Energía desolvatación	2(E y R)	40(20+,20-)	80
	Energía electroestática	2(E y R)	40(20+,20-)	80
Receptor	Energía desolvatación	2(E y R)	40(20+,20-)	80
	Energía electroestática	2(E y R)	40(20+,20-)	80
Ligando-Receptor	Contactos energía electroestática	3(E y R y R)	40(20+,20-)	120
	Contactos libres de la energía	3(E y R y R)	40(20+,20-)	120
Clases				1
Total de características				643

Tabla 5.2: Matriz $M1$, tipos de energías y características calculadas por FastContact para cada complejo [14].

5.3.5. Creación de la segunda Matriz

En la investigación desarrollada por [15], en consideración que el algoritmo FastContact sólo entregaba los 20 valores que más contribuyen y los 20 valores que menos contribuyen de cada tipo de energía, esto motivo a explorar el total de características que puede entregar FastContact por cada complejo de proteínas, pudiendo utilizar esa información para obtener mejores resultados en la clasificación.

Como primera aproximación se determinó que la lista de complejos puede tener menos de 40 valores energéticos (ver Tabla 5.3 en donde se muestra que existe al menos un complejo con 21 valores energéticos). Entonces en [15] se menciona que FastContact en lugar de estar

entregando 40 valores energéticos, puede estar entregando sólo los 21 valores energéticos y añade valores perdidos, o que los 19 valores energéticos faltantes están siendo repetidos, por lo cual se estudiaron los casos de superposición (Overlap) que se pueden producir en los valores energéticos.

Cadena	Tipo de energía	Min	Max	Usados
Complejo	Energía libre de unión	29	1.520	29
Ligando	Energía desolvatación	29	1.520	21
	Energía electroestática	29	1.520	21
Receptor	Energía desolvatación	21	1.475	21
	Energía electroestática	21	1.475	21
Receptor-Ligando	Contactos energía electroestática	2.064	129.999	268
	Contactos libres de la energía	2.064	129.999	268

Tabla 5.3: Mínimo y máximo número de valores energéticos obtenido desde 298 complejos [15].

5.3.5.1. Matriz 2

Para la creación de la Matriz M_2 se basó en el estudio de 298 complejos, los cuales se diferencian 94 complejos permanentes y 204 complejos transitorios. Finalmente la Matriz M_2 queda representada con una dimensión de 298x1.836 en la Ecuación 5.3.

$$M_2 = \begin{bmatrix} c_1 & y_{1,1} & y_{1,2} & \cdots & y_{1,1,836} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_1 & y_{94,1} & y_{94,2} & \cdots & y_{94,1,836} \\ c_2 & y_{95,1} & y_{95,2} & \cdots & y_{95,1,836} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_1 & y_{298,1} & y_{298,2} & \cdots & y_{298,1,836} \end{bmatrix} \tag{5.3}$$

Los valores de la Matriz M_2 fueron obtenidos de una versión personalizada de FastContact, lo que implica que en lugar de los primeros 20 valores energéticos positivos y negativos el algoritmo personalizado entrega de forma arbitraria el números de energías, utilizando para esto los rango límites de las características energéticas disponibles Tabla 5.3. El criterio

usado para el algoritmo personalizado es seleccionar las características que tengan el mínimo número de valores energéticos disponibles, es decir, elegir el mínimo número para tener el mismo número de características por complejo.

En la Tabla 5.4 se proporciona una referencia para los tipos de energía y las características disponibles para la Matriz M_2, en la tabla se puede observar que en el caso de energía libre de unión de la cadena Complejo se optó por utilizar las 29 mejores características, las cuales corresponden al mínimo numero de características disponible por cada complejo. En el caso de la cadena Ligando y la cadena Receptor el número de características debió ser igualado ya que las proteínas interactúan entre sí, para esto se eligió el valor mínimo disponible entre todos ellos en este caso el mínimo está entre 21 y 29. Finalmente para la cadena Ligando-Receptor se consideró el número mínimo de valores disponibles distintos de cero que fueran positivos o negativos.

			(20+, 20- Características)	
Cadena	Tipo de energía	Energía,residuos	Matriz 2	Total
Complejo	Energía electroestática			1
	Energía desolvatación			1
	Energía libre de unión	2(E y R)	29(14+,15-)	58
Ligando	Energía desolvatación	2(E y R)	21(10+,11-)	42
	Energía electroestática	2(E y R)	21(10+,11-)	42
Receptor	Energía electroestatica	2(E y R)	21(10+,11-)	42
	Energía electroestática	2(E y R)	21(10+,11-)	42
Ligando-Receptor	Contactos energía electroestática	3(E y R y R)	268(134+,134-)	804
	Contactos libres de la energía	3(E y R y R)	268(134+,134-)	804
Clases				1
Total de características				1837

Tabla 5.4: Matriz M_2, tipos de energías y características calculadas por FastContact personalizado para cada complejo.

A partir de la Matriz M_2 se construyeron tres conjuntos de datos que serán descritos a continuación:

a) Una matriz con sólo energías negativas: Solamente se consideran las energías que contribuyen más a la interacción con un total de 1.837 características incluyendo las características de clase.

De aquí en adelante este conjunto será reconocido como Top(-).

b) Una matriz con sólo energías positivas: Solamente se consideran energías que contribuyen menos a la interacción con un total de 1.837 características incluyendo la clase. De aquí en adelante este conjunto será reconocido como Top(+).

c) Una matriz con ambas energías negativas y positivas: El número de las energías es proporcional al tamaño del complejo, considerando las energías que contribuyen mas o menos a la interacción, con un total de 1.837 características incluyendo la clase. Esta matriz se creó en base a la división de el número de valores energéticos a la mitad, es decir, si el límite de los valores energéticos era 29 entonces podríamos seleccionar los mejores 15 valores negativos y los mejores 14 valores positivos. Se escogen más valores negativos por sobre los positivos, por que contribuyen más a la interacción. De aquí en adelante este conjunto será reconocido como Top(-)(+).

5.3.6. Creación de las Matrices sin residuos

Desde la creación de las Matrices M_1 y M_2 mencionadas en la sección anterior, se exploraron nuevas opciones de conjuntos de datos dando como resultado final nuevos conjuntos que sólo contendrán como características las energías (E) entregadas por la aplicación FastContact, es decir, que se suprimirán los residuos (R) y sólo se clasificará en base a las características de energía (E). A continuación se explicaran las dimensiones de estos nuevos conjuntos de datos.

Para la Matriz M_1 el cual contenía 282 características de energía (E) y 360 características de residuo (R), se creo un nuevo conjunto sin residuos, que finalmente tiene una dimensión representada por 296x282 (ver la Ecuación 5.4).

$$M_1^{'} = \begin{bmatrix} c_1 & x_{1,1} & x_{1,2} & ... & x_{1,282} \\ ... & ... & ... & ... & ... \\ c_1 & x_{93,1} & x_{93,2} & ... & x_{93,282} \\ c_2 & x_{94,1} & x_{94,2} & ... & x_{94,282} \\ ... & ... & ... & ... & ... \\ c_2 & x_{296,1} & x_{296,2} & ... & x_{296,282} \end{bmatrix} \tag{5.4}$$

Para la Matriz M_2 que se ve representada por tres conjuntos de datos mencionados anteriormente como Top(-), Top(+) y Top(-)(+), en donde cada uno de estos conjuntos tienen la misma dimensión de 651 características de energía (E) y 1.185 características de residuo (R). Entonces para cada conjunto de la Matriz M_2 se creo uno nuevo que no contenga residuos los cuales poseen la misma dimensión 298x651 (ver la Ecuación 5.5).

$$M_2^{'} = \begin{bmatrix} c_1 & y_{1,1} & y_{1,2} & \cdots & y_{1,651} \\ ... & ... & ... & ... & ... \\ c_1 & y_{94,1} & y_{94,2} & \cdots & y_{94,651} \\ c_2 & y_{95,1} & y_{95,2} & \cdots & y_{95,651} \\ ... & ... & ... & ... & ... \\ c_1 & y_{298,1} & y_{298,2} & \cdots & y_{298,651} \end{bmatrix} \tag{5.5}$$

Finalmente desde la Figura 5.4 se puede observar los ocho conjuntos obtenidos desde las Matrices originales M_1 y M_2, estos conjuntos serán la base para aplicación de los algoritmos de selección que se describirán en la siguiente sección. Los conjuntos de datos denominados Conjunto(20+,20-), Top(-), Top(+) y Top(-)(+) que se encuentran en el segundo nivel son los que se obtuvieron desde las investigación de [14] y [15]. Los conjuntos de datos sin residuo denominados ConjuntoE(2+,20-), TopE(-), TopE(+) y TopE(-)(+) que se encuentran en el último nivel fueron propuestos para poder explorar nuevas opciones de clasificación al utilizar sólo las características de energía (E).

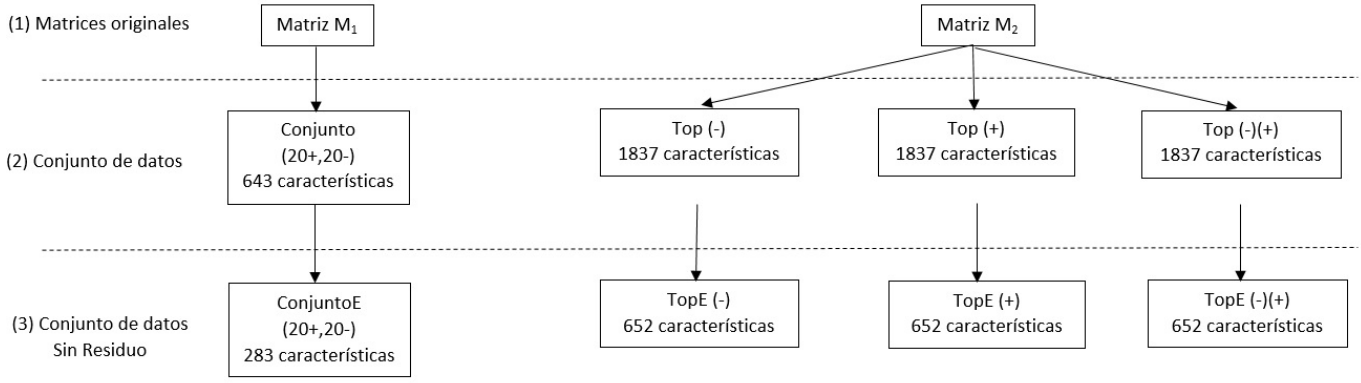


Figura 5.4: Conjuntos de datos finales.

5.4. Selección de Características

En la etapa anterior se logró obtener el espacio de características disponibles que se ven representadas a través de dos Matrices originales M_1 y M_2, estas matrices fueron creadas a partir de dos puntos de vista diferentes los cuales influyen en la cantidad de características que posee cada matriz.

Se ocuparán algoritmos de selección de características sobre los conjuntos de datos obtenidos desde M_1 y M_2 para poder obtener un subconjunto en donde se obtengan las características más relevantes y descarte las redundantes e irrelevantes. El objetivo de un algoritmo de selección es reducir la dimensión del conjunto original pero sin perder la exactitud del conjunto original, cuando se desea descartar una característica se escoge la que peor representa al conjunto.

Para el desarrollo de esta investigación se requiere obtener un subconjunto de características que permitan evaluar las separaciones entre las clases. Se implementaron tres algoritmos de selección de características, los cuales utilizan como criterio de evaluación la distancia de Chernoff, que permite medir la separabilidad de las clases.

Se escogieron los siguientes tres algoritmos (1) Sequential Forward Search (SFS), (2) Sequential Backward Search (SBS) y (3) Sequential Floating Forward Search (SFFS) los cuales pertenecen a la estrategia de búsqueda heurística. En comparación con los algoritmos de estrategia de búsqueda completa; los algoritmos heurísticos son más rápido en tiempo de ejecución y dada la cantidad de características energéticas de las matrices una búsqueda completa para encontrar el subconjunto óptimo podría llevar demasiado tiempo y consumir muchos recursos computacionales. Una de las razones principales por la cual no se eligieron algoritmos aleatorios, como son los algoritmos genéticos, es porque para la investigación se necesita comparar con los datos anteriormente obtenidos en Gutiérrez-Bunster [14] y los algoritmos aleatorios generan un resultado diferente cada vez que se ejecutan obteniéndose más de una posible solución. A continuación se explicarán el criterio de evaluación y los algoritmos seleccionados.

5.4.0.1. Distancia de Chernoff

Para los algoritmos seleccionados se utilizará la distancia de Chernoff como criterio de separabilidad. La distancia de Chernoff se realiza entre dos clases en este caso las IPP permanente e IPP transitorio. El criterio se define por la Ecuación 5.6:

$$FOC = \frac{p_1 p_2}{2}(m_1 - m_2)^t S_w^{-1}(m_1 - m_2) + \frac{1}{p_1 p_2}\log\left(\frac{|S_w|}{|S_1|^{p_1}|S_2|^{p_2}}\right) \tag{5.6}$$

Para poder utilizar la ecuación se identificaron el número total de complejos (n), el número de complejos permanentes ($n1$) y el número de complejos transitorios ($n2$), con lo cual se pueden obtener las probabilidades de cada clase $p1 = \frac{n_1}{n}$ y $p2 = \frac{n_2}{n}$. También se utilizan dos matrices X_1 y X_2, donde X_1 representa los complejos permanentes definidos por la clase 1 por el número de características de la iteración y X_2 representa los complejos transitorios definidos por la clase 2 por el número de características de la iteración. El número de características es determinado por la cantidad de iteraciones que ha realizado el algoritmo de selección inicialmente se comienza con una característica.

Con las matrices X_1 y X_2 se obtienen la covarianza por clase en S_1 y S_2, además con las mismas matrices se obtiene la media por clase como m1 y m2 respectivamente. Para obtener S_w se aplica la función $S_w = p_1 * S_1 + p_2 * S_2$ y para calcular los determinantes de S_1, S_2 y S_w se aplica los conocimiento de vectores y valores propios los cuales se define que la multiplicación de los valores propios es igual a la determinante de una matriz que queda definido por la Ecuación $|S| = \prod_{i=1}^{d} \lambda_i$,donde λ son los valores propios, esto implica que la covarianza sea Ecuación 5.7

$$S = \phi \Lambda \phi^t \tag{5.7}$$

En la Ecuación 5.7 de define S para cada S_1, S_1 y S_w, en donde ϕ es la matriz de vectores propios de S, Λ es la matriz de los valores propios de S, y ϕ^t es la matriz transpuesta de ϕ. Para calcular la inversa de S_w, se definió en la Ecuación 5.8

$$S_w^{-1} = (\phi\Lambda\phi^t)^{-1} \tag{5.8}$$

Para poder utilizar los valores propios que presentan ceros en la diagonal y evitar valores infinitos en [14] se estableció un umbral de $1,0x10^{-6}$, permitiendo manejar las matrices y obtener la inversa de S_w y evitar la singularidad. Además, se debe considerar que en primera instancia cuando las matrices sean muy pequeñas en algún momento puede que no existan valores propios que cumplan con el umbral mínimo en ese caso se utilizan todos los valores propios. Con este umbral se pretende reducir el tamaño de la matriz de vectores propios, eliminando las filas y columnas que no cumplan con el mínimo del umbral para la manipulación más eficiente de la matriz de vectores propios.

Finalmente se puede obtener el valor del criterio, las formulas se redujeron a operaciones simples evitando que pudiera ocurrir valores infinitos, logaritmos negativos o infinitos y matrices singulares. Este criterio aplicado a los algoritmos de selección de características, se considera el valor máximo obtenido y se compara con el nuevo, el que resulte mayor queda como valor máximo lo que irá cambiando hasta finalizar obteniendo las características ordenadas de acuerdo a las características que mas contribuyen a la interacción.

5.4.1. Búsqueda Secuencial Hacia Adelante (Sequential Forward Search - SFS)

El Algoritmo Búsqueda Secuencial Hacia Adelante (SFS) se caracteriza por comenzar con un conjunto vació de características $X(\emptyset)$ en el cual se irán agregando las mejores características según el criterios de evaluación utilizado $J(X)$, como entrada para este algoritmo se necesitan todas las características del conjunto $Y = (y_1, y_2, ...y_d)$, donde d es la cantidad total de características. El algoritmo se explica a continuación:

a) El algoritmo comienza con $X_0 = (\emptyset)$ y $k = 0$ seleccionando la mejor característica en donde se cumpla que $x^+ = argmax\, J(x_k + x)$, donde $x \in Y - X_k$, x^+ representa a la característica seleccionada y x_k representa al subconjunto.

b) Luego de elegida la característica x^+ se debe ingresar al subconjunto representado por X_k.

c) Finalmente se debe iterar y volver a hacer todos los pasos previos hasta alcanzar a satisfacer el criterio el cual puede ser un número preestablecido de características.

La salida de este algoritmo es $X_k = (x_1, x_2, ...x_p)$ donde p es la cantidad de características del subconjunto final.

5.4.2. Búsqueda Secuencial Hacia Atrás (Sequential Backward Search - SBS)

El Algoritmo Búsqueda Secuencial Hacia Atrás (SBS) se caracteriza por comenzar con el conjunto completo de características $X(Y)$ en el cual se irán sacando las características que peor contribuyen según el criterio de evaluación $J(X)$, como datos de entrada para este algoritmo se necesitan todas las características del conjunto $Y = (y_1, y_2, ...y_d)$, donde d es la cantidad total de características. El algoritmo se explica a continuación:

a) El algoritmo comienza con $X_0 = (Y)$ y $k = d$, en donde se ira eligiendo la peor característica la cual cumpla con $x^- = argmax J(x_k - x)$, donde $x \in X_k$, x^- representa a la característica seleccionada para eliminar y x_k representa al subconjunto.

b) Luego la característica x^- debe ser eliminada del subconjunto X_k.

c) Finalmente se debe iterar y volver a hacer todos los pasos previos hasta alcanzar a satisfacer el criterio el cual puede ser un número preestablecido de características.

La salida de este algoritmo es $X_k = (x_1, x_2, ...x_p)$ donde p es la cantidad de características del subconjunto final.

5.4.3. Búsqueda Flotante Secuencial Hacia Adelante (Sequential Floating Forward Search - SFFS)

Los Algoritmos Flotantes fueron definidos por Pudil et.al [38], esto motivado por el problema en que una característica no puede volver a hacer elegida una vez que es eliminada. La base de este estudio se realizo en el Algoritmo (L,R) el cual se caracteriza por agregar L veces y eliminar una cierta cantidad R de características, ellos argumentan que la dimensionalidad de las etapas se fija en función de los valores preestablecidos de L y R, argumentando que desafortunadamente no existe manera teórica de la predicción de los valores L y R para lograr el mejor conjunto de características. Es por esto que ofrecen una alternativa que en vez de fijar los valores previamente, estos valores floten para mantener la flexibilidad de cambiar con el fin de aproximar a la solución óptima lo mas cercano posible. El Algoritmo Búsqueda Flotante Secuencial Hacia Adelante (SFFS) es un algoritmo flotante en donde se caracteriza por comenzar con un conjunto vacío de características $X(\emptyset)$ en el cual se irán agregando las mejores características según el criterio de evaluación utilizado $J(X)$, como entrada para este algoritmo se necesitan todas las características del conjunto $Y = (y_1, y_2, ...y_d)$, donde d es la cantidad total de características. El algoritmo SFFS tiene la singularidad de poder eliminar características una vez que se agregan al subconjunto X con la idea de poder generar nuevos caminos y ampliar la evaluación del espacio de características. La eliminación de las características no es arbitraria debe cumplir con una condicionalidad que sera mencionada a continuación en la explicación del algoritmo:

a) El algoritmo comienza con $X_0 = (\emptyset)$ y $k = 0$ seleccionando la mejor característica en donde se cumpla que $x^+ = argmax\, J(x_k + x)$, donde $x \in Y - X_k$, x^+ representa a la característica seleccionada y x_k representa al subconjunto.

b) Luego de elegida la característica x^+ se debe ingresar al subconjunto representado por X_k

c) Una vez seleccionada la característica se debe aplicar la condicional de exclusión la cual es evaluar el subconjunto X con $x^- = argmax\, J(x_k - x)$. Si el elegido x^+ sigue siendo el mejor se debe volver a la etapa (a) pero si existe en el subconjunto X de la etapa anterior que cumpla con una combinación mejor se debe elegir la nueva x^- y eliminar la x^+.

d) Finalmente se debe iterar y volver a hacer todos los pasos previos hasta alcanzar a satisfacer el criterio el cual puede ser un número preestablecido de características.

La salida de este algoritmo es $X_k = (x_1, x_2, ...x_p)$ donde p es la cantidad de características del subconjunto final.

En el siguiente Capítulo se presentarán la implementación y aplicación de los algoritmos aquí presentados además de recalcar el uso de la distancia de Chernoff como criterio de evaluación entre las dos clases IPP permanentes e IPP transitorias. Finalmente se mostrarán los subconjuntos adquiridos por la utilización de cada uno de los algoritmos.

Capítulo 6

Implementación y Aplicación

La etapa de implementación se llevo a cabo en lenguaje de programación Python. Los algoritmos secuenciales fueron probados con los diferentes conjuntos nombrados en el capítulo anterior. En las siguientes secciones se presentará su implementación y la aplicación sobre los conjuntos de datos obteniendo como resultado de salida un ranking de las características ordenadas por su orden de importancia según el algoritmo.

Para la realización de los algoritmos se utilizó como criterio de evaluación la distancia de Chernoff que mide las distancia entre las clases, para diferenciar una clase de otra se debió realizar a la matriz original (ver Ecuación 6.1) una división en donde la clase 1 hace referencia a las características del complejo permanente c_1 (ver Ecuación 6.2) y la clase 2 hace referencia a las características del complejo transitorio c_2 (ver Ecuación 6.3), la dimensión de cada clase es definida en función de d lo cual señala la cantidad de características que existen en cada iteración del algoritmo.

$$M_{original} = \begin{bmatrix} c_1 & y_1 & y_2 & \dots & y_d \\ c_1 & y_1 & y_2 & \dots & y_d \\ \dots & \dots & \dots & \dots & \dots \\ c_2 & y_1 & y_2 & \dots & y_d \\ c_2 & y_1 & y_2 & \dots & y_d \end{bmatrix} \tag{6.1}$$

$$M_{clase1} = \begin{bmatrix} c_1 & y_1 & y_2 & \cdots & y_d \\ c_1 & y_1 & y_2 & \cdots & y_d \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_1 & y_1 & y_2 & \cdots & y_d \end{bmatrix} \tag{6.2}$$

$$M_{clase2} = \begin{bmatrix} c_2 & y_1 & y_2 & \cdots & y_d \\ c_2 & y_1 & y_2 & \cdots & y_d \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_2 & y_1 & y_2 & \cdots & y_d \end{bmatrix} \tag{6.3}$$

6.1. Implementación de los Algoritmos Secuenciales

A continuación se muestra la implementación de los algoritmos en donde se señala los requisitos de entrada para que se ejecute cada algoritmo, además del conjunto de salida del algoritmo.

6.1.1. Implementación Sequential Forward Search (SFS)

Algoritmo Sequential Forward Search
Entrada : Recibe la matriz completa de características **M**
Salida : Entrega una matriz de características ordenada **X** según el criterio de evaluación
1: Se inicializa la Matriz del subconjunto vacía **X**=$\emptyset$.
2: **Para** cada columna en la matriz se debe iterar hasta cumplir con la cantidad del subconjunto.
3: Se almacena en r la función criterio de evaluación sobre el subconjunto inicial $\mathbf{X} + \mathbf{M}[0]$.
4: **Para** cada valor de $i = 2, 3, .., n$, donde n es la cantidad de columnas de **M** se debe iterar
5: Se almacena en p la función criterio de evaluación sobre el subconjunto $\mathbf{X} + \mathbf{M}[i]$.
6: **Si** $p > r$ **Entonces**
7: r guarda el valor de p
8: Se agrega a X la mejor característica escogida r
9: Se elimina de M la característica escogida r
10: **Devuelve** X

Tabla 6.1: Implementación Sequential Forward Search.

6.1.2. Implementación Sequential Backward Search (SBS)

La implementación del algoritmo Sequential Backward Search (SBS) es muy similar a la SFS pero con la distinción que empieza a realizar la búsqueda con el conjunto completo de características y elimina la peor en cada iteración.

Algoritmo Sequential Backward Search
Entrada : Recibe la matriz completa de características **M**
Salida : Entrega una matriz de características ordenada **X** según el criterio de evaluación
1: Se inicializa la Matriz del subconjunto **X**=**M**.
2: **Para** cada columna en la matriz se debe iterar hasta cumplir con la cantidad del subconjunto.
3: Se almacena en r la función criterio de evaluación sobre el subconjunto inicial $\mathbf{X} - \mathbf{X}[n]$.
4: **Para** cada valor de $i = n-1, n-2, ..., 1$, donde n es la cantidad de columnas de **X** se debe iterar
5: Se almacena en p la función criterio de evaluación sobre el subconjunto $\mathbf{X} - \mathbf{X}[i]$.
6: **Si** $p > r$ **Entonces**
7: r guarda el valor de p
8: Se elimina de X la característica peor evaluada r
9: **Devuelve** X

Tabla 6.2: Implementación Sequential Backward Search.

6.1.3. Implementación Sequential Floating Forward Search (SFFS)

Algoritmo Sequential Floating Forward Search

Entrada : Recibe la matriz completa de características **M** y k que es el número de características

Salida : Entrega una matriz de características ordenada **X** según el criterio de evaluación

1: Se inicializa la Matriz del subconjunto vacía **X**=$\emptyset$.

2: **Para** cada columna en la matriz se debe iterar hasta cumplir con la cantidad del subconjunto.

3: **Si** la cantidad de columnas de $M > 0$

4: Se almacena en x^+ la función criterio de evaluación sobre el subconjunto inicial $\mathbf{X} + \mathbf{M}[0]$.

5: **Para** cada valor de $i = 2, 3, .., n$, donde n es la cantidad de columnas de **M** se debe iterar

6: Se almacena en p la función criterio de evaluación sobre el subconjunto $\mathbf{X} + \mathbf{M}[i]$.

7: **Si** $p > x^+$ **Entonces**

8: x^+ guarda el valor de p

9: Se agrega a X la mejor característica escogida x^+

10: Se elimina de M la característica escogida x^+

11: **Si** la cantidad de columnas de **X** >= 2

13: **Para** cada valor de $i = n-1, n-2, ..., 1$, donde n es la cantidad de columnas de **X**

14: Se almacena en p la función criterio de evaluación sobre el subconjunto $\mathbf{X} - \mathbf{X}[i]$.

15: **Si** $p > x^-$ **Entonces**

16: x^- guarda el valor de p

17: **Si** existe peor característica

18: Se elimina de X la característica peor evaluada x^-

19: **Devuelve** X

Tabla 6.3: Implementación Sequential Floating Forward Search.

6.2. Aplicación de los algoritmos sobre las matrices de características

Para aplicar los algoritmos desarrollados se debe leer desde un archivo en formato de texto, en donde se almacena las características de los conjuntos de datos mencionados en el Capítulo 4.

Los conjuntos en los que fueron aplicados los algoritmos de selección de características fueron sobre el Conjunto(20+,20-) y sin residuos ConjuntoE(20+,20-), pertenecientes a la Matriz M_1 (ver Ecuación 5.2) y los conjuntos sin residuos TopE(-), TopE(+) y TopE(-)(+), pertenecientes a la Matriz M_2 (ver Ecuación 5.3). Esta decisión fue tomada debido a las complicaciones de recursos computacionales y el tiempo de ejecución presentadas sobre los conjuntos de datos con 1.837 características un ejemplo concreto de esto fue la utilización del Algoritmo Sequential Backward Search (SBS) que dada su naturaleza es comenzar con el conjunto completo de características y necesita un tiempo de computo mayor por iteración, ya que comienza evaluando conjuntos del orden de 1.835 características, esto tardaba en promedio 2,5 horas para la exclusión de una característica. Es por esto que se utilizaron los conjuntos de datos que presentan una menor cantidad de características para alcanzar a evaluar los resultados en la clasificación.

En esta sección se muestran las primeras 20 características más discriminantes para la separabilidad de las clases utilizando como criterio de evaluación la distancia de Chernoff y algún algoritmo de selección de característica.

6.2.1. Aplicación sobre Conjunto(20+,20-)

En la Tabla 6.4 se puede observar las primeras 20 características utilizando el algoritmo Sequential Forward Search sobre el Conjunto(20+,20-), obtenido desde la Matriz M_1.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	577	**6**	356	**11**	313	**16**	293
2	27	**7**	271	**12**	306	**17**	319
3	275	**8**	326	**13**	244	**18**	362
4	202	**9**	311	**14**	418	**19**	337
5	248	**10**	512	**15**	344	**20**	299

Tabla 6.4: Lista de las primeras 20 características sobre el Conjunto(20+,20-) utilizando algoritmo SFS.

En la Tabla 6.5 se puede observar las primeras 20 características utilizando el algoritmo Sequential Backward Search sobre el Conjunto(20+,20-), obtenido desde la Matriz M_1.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	463	**6**	396	**11**	469	**16**	442
2	339	**7**	516	**12**	511	**17**	505
3	250	**8**	332	**13**	509	**18**	527
4	484	**9**	481	**14**	280	**19**	346
5	336	**10**	358	**15**	439	**20**	350

Tabla 6.5: Lista de las primeras 20 características sobre el Conjunto(20+,20-) utilizando algoritmo SBS.

En la Tabla 6.6 se puede observar las primeras 20 características utilizando el algoritmo Sequential Floating Forward Search sobre el Conjunto(20+,20-), obtenido desde la Matriz M_1.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	577	**6**	356	**11**	313	**16**	293
2	27	**7**	271	**12**	306	**17**	319
3	275	**8**	326	**13**	244	**18**	362
4	202	**9**	311	**14**	418	**19**	337
5	248	**10**	512	**15**	344	**20**	299

Tabla 6.6: Lista de las primeras 20 características sobre el Conjunto(20+,20-) utilizando algoritmo SFFS.

6.2.2. Aplicación sobre conjuntos sin residuos

En la Tabla 6.7 se puede observar las primeras 20 características utilizando el algoritmo Sequential Forward Search sobre el ConjuntoE(20+,20-), obtenido desde la Matriz M_1.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	62	**6**	1	**11**	122	**16**	121
2	85	**7**	40	**12**	201	**17**	37
3	22	**8**	42	**13**	81	**18**	282
4	41	**9**	202	**14**	38	**19**	120
5	2	**10**	39	**15**	199	**20**	82

Tabla 6.7: Lista de las primeras 20 características sobre el ConjuntoE(20+,20-) utilizando algoritmo SFS.

En la Tabla 6.8 se puede observar las primeras 20 características utilizando el algoritmo Sequential Forward Search sobre el conjunto TopE(-), obtenido desde la Matriz M_2.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	52	**6**	31	**11**	115	**16**	27
2	30	**7**	29	**12**	73	**17**	4
3	1	**8**	95	**13**	28	**18**	75
4	2	**9**	74	**14**	3	**19**	384
5	53	**10**	32	**15**	72	**20**	55

Tabla 6.8: Lista de las primeras 20 características sobre el conjunto TopE(-) utilizando algoritmo SFS.

En la Tabla 6.9 se puede observar las primeras 20 características utilizando el algoritmo Sequential Forward Search sobre el conjunto TopE(+), obtenido desde la Matriz M_2.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	1	**6**	651	**11**	52	**16**	27
2	2	**7**	114	**12**	29	**17**	68
3	31	**8**	73	**13**	28	**18**	94
4	115	**9**	71	**14**	113	**19**	72
5	30	**10**	112	**15**	49	**20**	111

Tabla 6.9: Lista de las primeras 20 características sobre el conjunto TopE(+) utilizando algoritmo SFS.

En la Tabla 6.10 se puede observar las primeras 20 características utilizando el algoritmo Sequential Forward Search sobre el conjunto TopE(-)(+), obtenido desde la Matriz M_2.

Posición	Característica	**Posición**	Característica	**Posición**	Característica	**Posición**	Característica
1	459	**6**	455	**11**	117	**16**	468
2	121	**7**	383	**12**	473	**17**	382
3	460	**8**	95	**13**	469	**18**	148
4	454	**9**	127	**14**	381	**19**	118
5	125	**10**	129	**15**	137	**20**	458

Tabla 6.10: Lista de las primeras 20 características sobre el conjunto TopE(-)(+) utilizando algoritmo SFS.

En la Tabla 6.11 se presenta un resumen de las combinaciones posibles entre los conjuntos obtenidos desde las Matrices M_1 y M_2 sobre los algoritmos de selección de características implementados (SFS, SBS y SFFS). Las casillas marcadas con ✓representan a los conjuntos que serán utilizados para la clasificación, entre estos se encuentran cuatro conjuntos sin ranking sin residuos y siete conjuntos con ranking que fueron obtenidos a través de la aplicación de un algoritmo de selección de características.

Conjuntos	Sin ranking	Con ranking		
		SFS	SBS	SFFS
Conjunto(20+,20-)		✓	✓	✓
Top(-)				
Top(+)				
Top(-)(+)				
ConjuntoE(20+,20-)	✓	✓		
TopE(-)	✓	✓		
TopE(+)	✓	✓		
TopE(-)(+)	✓	✓		

Tabla 6.11: Conjuntos que serán evaluados por el clasificador

Capítulo 7

Resultados

Para la clasificación de los conjuntos de datos se utilizará la aplicación Weka [16] desde la cual se han seleccionado dos algoritmos de clasificación (1) Maquina Soporte de Vectores (SVM) (al cual se aplicarán diferentes kernels tales como lineal, polinomial 2, polinomial 3, radial y sigmoide) y el algoritmo de clasificación (2) Bosques Aleatorios (RM). Para poder evaluar la clasificación se usará dos métodos conocidos como validación cruzada y división porcentual para lo que se utilizarán diferentes tamaños de conjuntos de entrenamiento y conjuntos de prueba [13], para encontrar la mejor clasificación. Los resultados obtenidos se trabajarán en términos de la precisión de la clasificación, es decir, el porcentaje total equivalente al 100 % menos el porcentaje de error obtenido. Por ejemplo, con un error del 25 %, se puede indicar que se posee un 75 % de precisión en la clasificación.

En la Tabla 7.1 se encuentra un resúmen de los resultados obtenidos sobre la clasificación de los diferentes conjuntos. En la primera columna se indican los conjuntos de datos que son utilizados para la clasificación y en las siguientes columnas se encuentran la precisión obtenida al utilizar los dos métodos de clasificación Máquina de soporte de vectores (SVM) y Bosques Aleatorios (RM).

[13] Resultados totales ver Apéndice E

Tabla 7.1: Mayores precisiones obtenidas desde los conjuntos de datos

Conjuntos de datos	Máquina de soporte de vectores					Bosques Aleatorios
	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
ConjuntoE(20+,20-)	72,64 %	80 %	73,03 %	75 %	71,43 %	**83,33 %**
TopE(-)	81,82 %	86,67 %	87,53 %	80 %	83,33 %	**87,88 %**
TopE(+)	**83,33 %**	79,41 %	79,07 %	77,52 %	**83,33 %**	76,67 %
TopE(-)(+)	76,40 %	81,82 %	75,50 %	79,19 %	83,33 %	**87,88 %**
Conjunto(20+,20-)SFS	75 %	72,57 %	69,20 %	72,28 %	72,28 %	**87,88 %**
Conjunto(20+,20-)SBS	73,33 %	82,14 %	74,58 %	77,28 %	72,28 %	**86,67 %**
Conjunto(20+,20-)SFFS	76,67 %	70,27 %	76,67 %	72,28 %	72,28 %	**80 %**
ConjuntoE(20+,20-)SFS	77,23 %	74,32 %	70,04 %	71,43 %	71,43 %	**82,14 %**
TopE(-)SFS	80 %	80 %	76,17 %	80 %	83,33 %	**86,11 %**
TopE(+)SFS	73,33 %	**84,85 %**	76,89 %	80 %	83,33 %	76,67 %
TopE(-)(+)SFS	71,81 %	70,47 %	71,14 %	80 %	83,33 %	**86,67 %**

Desde una perspectiva global se puede apreciar que el clasificador Bosques Aleatorios (RM) presenta mejor precisión en la mayoría de los conjuntos de datos, excepto TopE(+) y TopE(+)SFS. De los conjuntos con los que fueron aplicados algoritmos de selección de características se destacan Conjunto(20+,20-)SFS, Conjunto(20+,20-)SBS, TopE(-)SFS y TopE(-)(+)SFS todos sobre 86 % de precisión.

Los conjuntos de datos a los cuales no se aplicaron algoritmos de selección de características como ConjuntoE(20+,20-) presenta mejor precisión con RM, TopE(-) obtiene mejor precisión con SVM kernel polinomial 3 y con RM, el conjunto TopE(+) obtiene una mayor precisión con SVM kernel lineal y kernel Sigmoid, finalmente TopE(-)(+) presenta mayor precisión con RM.

De los conjuntos de datos pertenecientes a la Matriz M_1 que se aplicaron los algoritmos de selección de características como son el Conjunto(20+,20-)SFS, Conjunto(20+,20-)SBS y Conjunto(20+,20-)SFFS presentaron una mayor precisión con RM.

Los conjuntos sin residuos al aplicar los algoritmos de selección de características presentan mejor precisión con RM como es ConjuntoE(20+,20-)SFS, TopE(+)SFS y TopE(-)(+)SFS, excepto TopE(+)SFS que al igual que el conjunto TopE(+) y en el trabajo de [15] los conjuntos con sólo energías positivas se comportan similar teniendo una atracción mayor a obtener mejor precisión con SVM.

Algunos de los conjuntos que presentan una menor precisión son Conjunto(20+,20-)SFFS, ConjuntoE(20+,20-)SFS y ConjuntoE(20+,20-) esto puede hacer referencia que los conjunto que presentan una disminución de características para la clasificación obtiene más bajos resultados, en estos casos en particular se parte del conjunto original Conjunto(20+,20-) el cual posee un conjunto de 642 características y al aplicar el algoritmo SFFS esta dimensión se disminuye a un conjunto de 393 características formando al Conjunto(20+,20-)SFFS. Lo mismo ocurre con los conjuntos sin residuo ConjuntoE(20+,20-)SFS y ConjuntoE(20+,20-) que disminuyen a un conjunto de 282 características cada uno.

En la Tabla 7.2 se observan las precisiones obtenidas por los clasificadores. Como se puede observar en la primera columna se presentan la cantidad de entrenamiento y prueba que fueron utilizados, esta sección se divide en dos en la primera se muestran las Validaciones Cruzadas en la formato n-pliegues, donde n es la cantidad de pliegues elegida y en la segunda se muestran la División Porcentual en formato n-m, donde n es el porcentaje de entrenamiento y m el porcentaje de prueba. Al final de la tabla se realiza un resumen de las precisiones más altas obtenidas según el método utilizado. Analizando la Tabla 7.2 se concluye que el método con el cual se obtienen mejor precisión en la clasificación es División Porcentual obteniendo en muchos casos sobre 80 %, independientemente del clasificador utilizado.

Tabla 7.2: Mayores precisiones obtenidas por clasificador

Entrenamiento	Máquina de soporte de vectores					Bosques
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	Aleatorios
2-pliegues	78,19 %	78,19 %	76,17 %	76,17 %	68,92 %	78,52 %
5-pliegues	78,52 %	78,52 %	76,51 %	78,86 %		80,54 %
8-pliegues	79,87 %	77,52 %	77,52 %	77,52 %		80,54 %
10-pliegues	81,54 %	79,19 %	78,52 %	77,52 %		81,21 %
12-pliegues	79,53 %	79,19 %	78,19 %	77,52 %		80,87 %
14-pliegues	79,87 %	79,53 %	78,86 %	77,52 %		81,21 %
15-pliegues	80,54 %	79,19 %	78,52 %	77,85 %		78,86 %
20-80		79,83 %	77,31 %			76,79 %
38-62	75,14 %	77,30 %		75,68 %		77,17 %
50-50		77,18 %		79,19 %		81,21 %
66-34	77,23 %	76,24 %	75,25 %	77,28 %	72,28 %	80,20 %
70-30	80,90 %	77,53 %	75,28 %	77,53 %		82,02 %
71-29	82,56 %	76,74 %	79,07 %			
74-26	77,92 %	80,52 %	75,32 %	79,22 %		80,52 %
78-22	81,82 %	81,82 %	84,85 %	78,79 %		84,85 %
80-20	80 %	80 %	83,33 %	76,67 %		83,33 %
81-19	80,70 %	82,14 %	82,46 %	77,19 %		84,21 %
84-16	77,08 %	83,33 %	**87,5 %**	77,08 %	77,08 %	83,33 %
86-14	**83,33 %**	78,57 %			78,57 %	
88-12	**83,33 %**	75 %		77,78 %	80,56 %	86,11 %
89-11	81,82 %	84,85 %	84,85 %	78,79 %	81,82 %	**87,88 %**
90-10	80 %	**86,67 %**	86,67 %	**80 %**	**83,33 %**	86,67 %
Validación Cruzada	81,54 %	79,53 %	78,86 %	78,86 %	68,92 %	81,21 %
División Porcentual	**83,33 %**	**86,67 %**	**87,5 %**	**80 %**	**83,33 %**	**87,88 %**

La Tabla 7.3 entrega las precisiones obtenidas dependiendo del método utilizado, para los conjuntos de datos sin ranking. Desde estos conjuntos se pueden destacar el conjunto TopE(-) y TopE(-)(+) con uno con precisión de 87,88 %, además de poseer sobre 9 métodos sobre el 80 %, la mayoría perteneciente a División Porcentual.

La Tabla 7.4 entrega las precisiones para los conjuntos de datos con ranking aplicados al Conjunto(20+,20-). Los algoritmos de selección de características que obtuvieron mejores resultados fueron SFS obteniendo una precisión de 87,88 % con División Porcentual 89-11 y SBS obteniendo una precisión de 86,67 % División Porcentual 90-10. Con la aplicación del algoritmo SBS se obtienen cuatro precisiones sobre el 80 %.

La Tabla 7.5 entrega las precisiones para los conjuntos de datos con ranking y sin residuos. Se pueden destacar TopE(-)SFS obteniendo una precisión de 86,11 % con División Porcentual 88-12 y TopE(-)(+)SFS obteniendo una precisión de 86,67 % con División Porcentual 90-10. Desde los resultados obtenidos desde la clasificación se puede hacer referencia que los conjuntos TopE(-) (solo con energías negativas) y TopE(-)(+) (combinación de energías positivas y negativas) en la mayoría de los casos obtienen mejor precisión que el conjunto TopE(+) (sólo con energías positivas).

Tabla 7.3: Mayores precisiones obtenidas por conjuntos de datos sin ranking

Entrenamiento Prueba	Conjuntos de datos			
	Sin Ranking			
	ConjuntoE(20+,20-)	TopE(-)	TopE(+)	TopE(-)(+)
2-pliegues	77,03 %	78,19 %	78,19 %	78,52 %
5-pliegues	78,04 %	78,52 %	77,18 %	78,86 %
8-pliegues	79,39 %	79,87 %	77,52 %	80,54 %
10-pliegues	76,01 %	81,54 %	77,18 %	81,21 %
12-pliegues	78,38 %	79,53 %	77,18 %	80,87 %
14-pliegues	79,05 %	79,87 %	77,52 %	81,21 %
15-pliegues	78,38 %	80,54 %	76,17 %	78,86 %
20-80	75,95 %	75,63 %	79,41 %	79,83 %
38-62	77,17 %	77,30 %		
50-50	76,35 %	78,52 %		81,21 %
66-34	78,22 %		77,23 %	77,23 %
70-30	78,65 %		80,90 %	78,65 %
71-29			82,56 %	
74-26	77,92 %	80,5195	79,2208	80,52 %
78-22	76,92 %	84,85 %		83,33 %
80-20	76,27 %	83,33 %	76,67 %	83,33 %
81-19	76,79 %	82,47 %	78,95 %	84,21 %
84-16		87,5 %		
86-14			**83,33 %**	
88-12			**83,33 %**	
89-11		**87,88 %**		**87,88 %**
90-10	**83,33 %**	86,67 %	**83,33 %**	83,33 %
Validación Cruzada	79,39 %	81,54 %	78,19 %	81,21 %
División Porcentual	**83,33 %**	**87,88 %**	**83,33 %**	**87,88 %**

Tabla 7.4: Mayores precisiones obtenidas por conjuntos de datos con ranking

Entrenamiento Prueba	Conjuntos de datos		
	Con Ranking		
	Conjunto(20+,20-)SFS	Conjunto(20+,20-)SBS	Conjunto(20+,20-)SFFS
2-pliegues	77,36 %	78,04 %	72,30 %
5-pliegues	79,05 %	76,69 %	72,97 %
8-pliegues	78,38 %	75,68 %	73,31 %
10-pliegues	77,70 %	76,69 %	71,96 %
12-pliegues	78,38 %	77,70 %	73,32 %
14-pliegues	76,01 %	78,04 %	75,34 %
15-pliegues	78,72 %	77,37 %	73,31 %
20-80	73,84 %	73,42 %	69,20 %
38-62	76,63 %	75 %	70,11 %
50-50	75 %	77,70 %	75 %
66-34	79,21 %	80,20 %	75,25 %
70-30	78,65 %	82,02 %	77,53 %
71-29			
74-26	79,22 %	77,92 %	77,92 %
78-22	76,92 %	75,38 %	72,31 %
80-20	79,66 %	77,97 %	71,19 %
81-19		82,14 %	75 %
84-16			
86-14			
88-12			
89-11	**87,88 %**		
90-10	83,33 %	**86,67 %**	**80 %**
Validación Cruzada	79,05 %	78,04 %	75,34 %
División Porcentual	**87,88 %**	**86,67 %**	**80 %**

Tabla 7.5: Mayores precisiones obtenidas por conjuntos de datos con ranking y sin residuos

Entrenamiento Prueba	Conjuntos de datos			
	Con Ranking			
	ConjuntoE(20+,20-)SFS	TopE(-)SFS	TopE(+)SFS	TopE(-)(+)SFS
2-pliegues	78,04 %	77,18 %	76,85 %	76,17 %
5-pliegues	78,72 %	77,52 %	78,86 %	80,54 %
8-pliegues	78,04 %	77,18 %	78,52 %	78,19 %
10-pliegues	78,04 %	78,19 %	78,19 %	79,53 %
12-pliegues	78,72 %	78,86 %	78,86 %	79,20 %
14-pliegues	77,36 %	77,18 %	79,19 %	80,20 %
15-pliegues	77,36 %	77,52 %	78,19 %	77,52 %
20-80	76,79 %	76,47 %	76,89 %	74,79 %
38-62	77,17 %	75,68 %		73,51 %
50-50	76,35 %	79,87 %	79,87 %	79,19 %
66-34	77,23 %	76,24 %	72,28 %	73,27 %
70-30	79,78 %		75,28 %	77,52 %
71-29				
74-26	76,62 %	79,22 %	79,22 %	75,32 %
78-22	76,92 %	80,30 %	78,79 %	78,79 %
80-20	77,97 %	80	76,67 %	78,33 %
81-19	82,14 %	80,70 %	77,19 %	80,78 %
84-16				
86-14				
88-12		**86,11 %**		
89-11			**84,85 %**	
90-10	**80 %**	83,33 %	83,33 %	**86,67 %**
Validación Cruzada	78,72 %	78,86 %	79,19 %	80,54 %
División Porcentual	**80 %**	**86,11 %**	**84,85 %**	**86,67 %**

7.1. Trabajos Futuros

Siguiendo la misma linea de investigación se puede seguir obteniendo conocimiento a través de los diferentes conjuntos que se ven señalados en la Tabla 7.6 los cuales están marcadas con el simbolo χ. Estos nuevos conjuntos que se pueden obtener al aplicar algoritmos de selección de características motivan a seguir trabajando para obtener nuevos resultados que pueden resultar en una precisión mayor a la presentada en esta investigación, además se puede seguir mejorando los algoritmos de selección de características, estudiando nuevos criterios de evaluación que sean aplicables en la separabiliadad de las clases para poder ejecutar los conjuntos que poseen 1837 características en un tiempo razonable.

Conjuntos	Con ranking		
	SFS	SBS	SFFS
Conjunto(20+,20-)			
Top(-)	χ	χ	χ
Top(+)	χ	χ	χ
Top(-)(+)	χ	χ	χ
ConjuntoE(20+,20-)		χ	χ
TopE(-)		χ	χ
TopE(+)		χ	χ
TopE(-)(+)		χ	χ

Tabla 7.6: Conjuntos futuros para investigación

Capítulo 8

Conclusiones

Las Interacciones Proteína-Proteína son estudiadas a través de los datos obtenidos desde las bases de datos de proteínas (PDB) otorgando la oportunidad de obtener la proteína representada en un formato tridimensional con la cual se hizo posible trabajar en esta investigación, dando una aproximación de como utilizar los algoritmos de selección de características sobre las propiedades energéticas que se encuentran en la superficie de interacción con el objetivo de clasificar las que corresponden a interacciones permanentes y transitorias. En la investigación se trabajó con diferentes conjuntos de datos los cuales fueron obtenidos a través de investigaciones previas, estos conjuntos estaban previamente clasificados en interacciones permanentes y transitorias. Desde estos conjuntos de datos además se crearon nuevos conjuntos los cuales sólo contenían como característica las energías sin residuo.

A través de la revisión bibliográfica del área se logró seleccionar tres algoritmos, Sequential Forward Search (SFS), Sequential Backward Search (SBS) y Sequential Floating Forward Search (SFFS) los cuales fueron desarrollados en lenguaje de programación Python para poder cumplir con los objetivos planteados. Estos algoritmos entregaron un conjunto de características las cuales se encuentran ordenadas por su contribución a la separabilidad entre clases, utilizando como criterio de evaluación la Distancia de Chernoff.

Para estudiar la precisión de la clasificación se utilizo la aplicación Weka desde la cual se utilizaron dos métodos Máquina de soporte de vectores (SVM) y Bosques aleatorios (RM), sobre estos métodos se aplicaron diferentes divisiones para obtener diferentes conjuntos de entrenamiento y conjuntos de prueba a través de Validación Cruzada y División Porcentual. Los valores más altos se lograron utilizando Bosques Aleatorios (RM) en combinación con División Porcentual obteniendo una precisión de 87,88 %.

En un futuro se puede buscar nuevas formas con las cuales se pretendan generar nuevas características que ayuden a mejorar la precisión en la clasificación. Además no se debe cerrar la posibilidad a generar nuevas investigaciones con nuevos algoritmos de selección y estudiar nuevos criterios de evaluación que puedan ser aplicados para medir la separabilidad de las clases.

Referencias

[1] Berggård, T., Linse, S., and James, P. Methods for the detection and analysis of protein-protein interactions. *Proteomics 7*, 16 (2007), 2833–2842.

[2] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. The protein data bank. *Nucleic Acids Research 28*, 1 (2000), 235–242.

[3] bionova. <http://www.bionova.org.es/biocast/tema08.htm> [Consulta: 20-06-2016].

[4] Breiman, L. Random Forests. *Machine learning 45.1* (2001), 5–32.

[5] Camacho, C. J., and Zhang, C. FastContact: Rapid estimate of contact and binding free energies. *Bioinformatics 21*, 10 (2005), 2534–2536.

[6] Dash, M., and Liu, H. Feature selection for classification. *Intelligent Data Analysis 1*, 3 (1997), 131–156.

[7] Decker, K. M., and Focardi, S. Technology overview: A report on data mining.

[8] Doak, J. An Evaluation of Feature Selection Methods and Their Application to Computer Security(Technical Report CSE-92-18).

[9] energia-nuclear.net. <http://energia-nuclear.net/definiciones/electron.html> [Consulta: 10-08-2016].

[10] Estructura Proteína. http://www.uaz.edu.mx/histo/TortorAna/ch02/02_22.jpg [Consulta: 20-06-2016].

[11] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. From Data Mining to Knowledge Discovery in Databases. *AI Magazine 17*, 3 (1996), 37.

[12] González M. <http://quimica.laguia2000.com/conceptos-basicos/cinetica-enzimaticaixzz4GW8hJjWi> [Consulta: 10-08-2016].

[13] González M. <http://quimica.laguia2000.com/conceptos-basicos/interacciones-hidrofobicasixzz4Gi56CMNj> [Consulta: 10-08-2016].

[14] Gutiérrez-bunster, T. Estudio de características energéticas en zonas de interacción proteína-proteína, para identificación de interacciones transitorias y permanentes.

[15] Gutiérrez-bunster, T., and Poo-caamaño, G. Improving Energetic Feature Selection to Classify Protein – Protein Interactions.

[16] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: An update. *SIGKDD Explorations 11*, 1 (2009).

[17] Herrera, F., and Cano, J. R. Técnicas de reducción de datos en KDD. El uso de Algoritmos Evolutivos para la Selección de Instancias. *Actas del I Seminario Sobre Sistemas Inteligentes (SSI'06), Universidad Rey Juan Carlos, Madrid (Spain).* (2006), 165–181.

[18] Hue, M., Riffle, M., Vert, J.-P., and Noble, W. S. Large-scale prediction of protein-protein interactions from structures. *BMC bioinformatics 11* (2010), 144.

[19] iupac. <http://goldbook.iupac.org/C01238.html> [Consulta: 10-08-2016].

[20] Jain, a., and Zongker, D. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, ...* (1997), 1–18.

[21] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: A review. *ACM Comput. Surv. 31*, 3 (1999), 264–323.

[22] Jiménez Montaño, M. Á. Reconocimiento De Patrones Analiss De Secuencias Moleculares E Informacion Biologica, 1989.

[23] Kudo, M., and Sklansky, J. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition 33*, 1 (2000), 25–41.

[24] Langley, P. Selection of Relevant Features in Machine Learning. *In Proceedings of the AAAI Fall Symposium on Relevance* (1994), 140–144.

[25] Leticia Maria. Reconocimiento de patrones utilizando técnicas estadısticas y conexionistas aplicadas a la clasificación de dígitos manuscritos.

[26] Luscombe, N. M., Greenbaum, D., and Gerstein, M. What is bioinformatics ? An introduction and overview. *Yearbook of Medical Informatics* (2001), 83–100.

[27] Marcey D. <http://gmein.uib.es/moleculas/fuerzas$_p$*roteinas*/*fuerzaproteinasjmol.html* >

[*Consulta* : 10 − 08 − 2016].

[28] Mckee, T., and Mckee, J. R. BIOQUÍMICA LAS BASES MOLECULARES DE LA VIDA. 123–154.

[29] Miles B. <https://www.tamu.edu/faculty/bmiles/lectures/electrontrans.pdf> [Consulta: 10-08-2016].

[30] Mintseris, J., and Weng, Z. Atomic Contact Vectors in Protein-Protein Recognition. *Proteins: Structure, Function and Genetics 53*, 3 (2003), 629–639.

[31] Mintseris J, W. Z. . <http://zlab.bu.edu/julianm/MintserisWengPNAS05.html> [Consulta: 23-06-2016].

[32] Nebil, O., and Can, T. Predicting Protein-Protein Interactions from Protein Sequences Using Phylogenetic Profiles.

[33] Nooren, I. M., and Thornton, J. M. NEW EMBO MEMBER'S REVIEW: Diversity of protein-protein interactions. *The EMBO Journal 22*, 14 (2003), 3486–3492.

[34] OpenCV. http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html [Consulta: 22-06-2016].

[35] Philippe J. <http://bioinformatics.oxfordjournals.org/content/18/suppl$_1$ /$S276.abstract$ > [$Consulta$: $10-08-2016$].

[36] Phizicky, E. M., and Fields, S. Protein-protein interactions: methods for detection and analysis. *Microbiological reviews 59*, 1 (1995), 94–123.

[37] Pliego P., Rodriguez E., Tetlalmatzi M. and Soto C. http://www.uaeh.edu.mx/scige/boletin/icbi/n1/e7.html [Consulta: 22-06-2016].

[38] Pudil, P., Novovicova, J., and Kittler, J. Floating search methods in feature selection. *Pattern Recognition* (1993).

[39] Pérez J. and Merino M. <http://definicion.de/puente-de-hidrogeno> [Consulta: 10-08-2016].

[40] Rao, V. S., Srinivas, K., Sujini, G. N., and Kumar, G. N. S. Protein-Protein Interaction Detection: Methods and Analysis. *International Journal of Proteomics 2014*, ii (2014), 1–12.

[41] Ruiz, R. Selección de Atributos mediante proyecciones. 180.

[42] Russis, V. D. L. T., Valles, A., Gómez, R., Chinea, G., and Pons, T. Interacciones proteína-proteina: bases de datos y métodos teóricos de predicción.

[43] Salah, S. A., Beltrán, N. H., Bustos, M. A., and Loyola, E. A. Selección de Características usando Algoritmos Genéticos para Clasificación de Vinos Chilenos Resumen. *Focus* (2007), 1–13.

[44] Schlimmer, J. C. Efficiently Inducing Determinations: A Complete and Systematic Search Algorithm that Uses Optimal Pruning. *Proceedings of the Tenth International Conference on Machine Learning*, 1987 (1993), 284–290.

[45] Valencia, P. E. Selección de características para clasificadores neuronales. *Anales del Instituto de Ingenieros de Chile*, July (1999), 65–74.

[46] Vázquez E. <http://laguna.fmedic.unam.mx/~evazquez/0403/estructura%20terciaria2a.html>
[Consulta: 10-08-2016].

Apéndice A

Definiciones

Reacción de condensación: Es una reacción química en la que dos moléculas se combinan para formar una molécula más grande, junto con la perdida de una molécula pequeña, que en la mayoría de los casos es una molécula de agua [19].

Enlaces entre los radicales R: Los enlaces entre los radicales R de los aminoácidos son aquellos que mantienen estable la conformación globular, la cual facilita la solubilidad en agua y así realizar funciones de transporte, enzimáticas, hormonales, etc. Las uniones entre los radicales R de los aminoácidos pueden darse por: (1)Los puentes de hidrógeno, (2) Puentes eléctricos, (3) Interacción Hidrófobas y (4) Puente disulfuro entre los radicales de aminoácidos que tiene azufre.

Puente de Hidrógeno: El puente de hidrógeno es una clase de enlace que se produce a partir de la atracción existente en un átomo de hidrógeno y un átomo de oxígeno, flúor o nitrógeno con carga negativa. Dicha atracción, por su parte, se conoce como interacción dipolo-dipolo y vincula el polo positivo de una molécula con el polo negativo de otra [39].

Puente Eléctrico: El puente eléctrico es más conocido como fuerzas de Van der Waals, estas fuerzas son atracciones eléctricas débiles entre diferentes átomos, que son el resultado de las fuerzas atractivas y repulsivas que se establecen al acercarse dichos átomos, de manera que existe una distancia en que la atracción es máxima (esta

distancia se conoce como radios de Van der Waals). Las fuerzas de Van der Waals se deben a que cada átomo puede tener un dipolo transitorio en un enlace y que en otro enlace puede incluir un dipolo complementario, provocando que dos átomos entre los diferentes enlaces se mantengan juntos. Estos dipolos transitorios provocan una atracción electrostática débil (fuerzas de Van der Waals). Estas atracciones de Van der Waals, aunque transitorias y débiles son un componente importante en la estructura de las proteínas porque existen muchas de ellas. La mayoría de los átomos de una proteína están empaquetados lo suficientemente próximos unos de otros para involucrar estas fuerzas transitorias [27].

Interacción Hidrófobas: La interacción hidrófoba se conoce comúnmente por tener la capacidad de no ser miscible con el agua, ni por puentes de hidrógeno ni mediante interacciones ion-dipolo. La interacción hidrófoba que se encuentran en el plegamiento de las proteínas, se producen cuando los aminoácidos con cadenas laterales no polares tienden a localizarse en el interior de la proteína, en donde se asocian con otras cadenas no polares, mientras que los aminoácidos polares suelen localizarse en la superficie de la proteína, para que la estructura resultante sea lo más estable posible [13].

Puente Disulfuro: El puente disulfuro es un enlace covalente formado por dos grupos sulfidrilo (-SH), cada uno de ellos perteneciente a un residuo de cisteína, se unen de manera covalente para formar un residuo de cistína. Los dos residuos que forman al puente, pueden estar separados por muchos aminoácidos en la secuencia o bien pueden pertenecer a diferentes cadenas polipeptídicas; el plegamiento de la(s) cadena(s) polipeptídicas, lleva a los residuos de cisteína a estar muy próximos, lo que permite la formación del enlace disulfuro. La formación de este enlace estabiliza la estructura tridimensional de la proteína [46].

Electrón: Un electrón es una partícula elemental estable cargada negativamente que constituye uno de los componentes fundamentales del átomo [9].

Transporte de electrones por citocromos: Los citocromos son proteínas que contienen grupos hemo prostéticos que funcionan como transportadores de electrones. Los

citocromos se pueden encontrar en la membrana celular de las bacterias, en la membrana interna de la mitocondria y de los clorolastos. Durante la respiración y fotosíntesis, las moléculas del citocromo aceptan y liberan alternativamente electrones [29].

Enzima: Las enzimas son biomoléculas especializadas en la catálisis de las reacciones químicas que tienen lugar en la célula. Son muy eficaces como catalizadores ya que son capaces de aumentar la velocidad de las reacciones químicas mucho más que cualquier catalizador artificial conocido, y además son altamente específicos ya que cada uno de ellos induce la transformación de un sólo tipo de sustancia y no de otras que se puedan encontrar en el medio de reacción.

Cinética enzimática: La cinética enzimática es la disciplina que estudia la velocidad en las reacciones químicas en las que intervienen enzimas. El estudio de esta velocidad y de la dinámica de la enzima, nos permite conocer a fondo el mecanismo de acción de dicha enzima, el rol que cumple en el metabolismo y la regulación de su actividad por inhibidores naturales, fármacos, venenos u otro tipo de sustancias [12].

Canalización de sustratos: La canalización de sustratos es el proceso de transferencia directa de un intermediario metabólico entre los sitios activos de dos enzimas que catalizan reacciones secuenciales en una ruta biosintética.

Especificidad de una proteína: La especificidad de las proteínas indica que cada una de ellas lleva a cabo una determinada función y lo realiza porque poseen una determinada estructura primaria y una conformación espacial propia. Las proteínas no son iguales en todos los organismos, cada individuo posee proteínas especificas, la semejanza entre proteínas son un grado de parentesco entre individuos, por lo que sirve para la construcción de arboles filogenéticos.

Perfiles filogenéticos (Phylogenetic Profiles): El perfil filogenético de una proteína es una cadena que codifica la presencia o ausencia de la proteína en cada genoma secuenciado en su totalidad. Debido a que las proteínas que participan en una ruta metabólica compleja o estructural común es probable que evolucionen de forma correlacionada, los perfiles filogenéticos de tales proteínas son a menudo similar o por lo

menos relacionados el uno al otro [35].

Cuando dos proteínas han sido seleccionadas evolutivamente para interaccionar, la mutación ventajosa en una de ellas tienden a seleccionar mutaciones en la otra proteína. Estudiando el perfil filogenético de dos proteínas se puede obtener información sobre el grado de similitud de su historia evolutiva. Cuanto más similar sea esta, más posibilidades existen de que esas dos proteínas interaccionen.

Cristalografía de rayos X: Es una forma de alta resolución de microscopia que permite la visualización de las estructuras de proteínas a nivel atómico para mejorar la comprensión de la interacción y función de las proteínas. Es posible ver a través de este método como la proteínas interactúan con otras moléculas y los cambios conformacionales de las enzimas.

Coinmunoprecipitación: Confirma las interacciones utilizando un extracto de células completas en donde las proteínas están presentes en su forma nativa.

Espectroscopia de resonancia magnética nuclear: Nace por la necesidad de analizar las Interacciones Proteína-Proteína mediante resonancia magnética nuclear (RMN). Dado que la superficie de unión es un aspecto crucial en el análisis de la interacción de proteínas, la espectroscopia de RMN se basa en que los núcleos magnéticos activos orientados por un fuerte campo magnético absorben la radiación electromagnética en la región de las frecuencias de radio que se rigen por su entorno químico.

Sistema de dos híbrido: Se lleva a cabo mediante la selección de una proteína de interés puesta a prueba contra una biblioteca aleatoria de potenciales proteínas para su interacción. Este análisis permite el reconocimiento directo de las Interacciones Proteína-Proteína, sin embargo implica una gran número de falsos positivos.

Enfoques expresión génica: Predice la interacción basada en la idea de que las proteínas que pertenecen a un perfil común son mas propensas a interactuar entre si que las proteínas que pertenecen a diferentes grupos.

Enfoques basados en la estructura: Predice las Interacciones Proteína-Proteína en base si dos proteínas tienen una estructura similar (primaria,secundaria o terciaria), por

ejemplo, si dos proteínas A y B pueden interactuar entre si, entonces puede existir otras dos proteínas cuyas estructuras son similares tal que A'y B' pueden interactuar entre si. Pero la mayoría de las proteínas pueden no poseer una estructura conocida lo que implica que el primer paso de este método es adivinar la estructura de la proteína en base a su secuencia.

Apéndice B

Recomendación de Algoritmo de Selección

Kudo y Sklansky [23] realizaron una aproximación de como poder escoger un algoritmo de selección dependiendo del (1) Tipo de objetivo,(2) Tamaño del conjunto de características y (3) Criterio de evaluación utilizado.

Desde la Tabla B.1 los tipos de criterios se ven representados por [A] Encontrar el mejor subconjunto para un determinado tamaño, [B] Encontrar el subconjunto mas pequeño que satisfaga una condición y [C] Encontrar un subconjunto cuya combinación de tamaño y tasa de error son óptimas. El tamaño del conjunto se ve diferenciado como pequeño ($n < 20$), mediano ($20 \leq n < 50$), grande ($50 \leq n \leq 100$) y muy grande ($100 < n$). Finalmente basándose en el criterio de evaluación se diferencian en monótono, aproximado y no monótono. A continuación se identifican los algoritmos utilizados en la Tabla.

SFFS: Búsqueda Secuencial Hacia Adelante (Sequential Forward Floating Search)

SBFS: Búsqueda Secuencial Hacia Atrás (Sequential Backward Floating Search)

GA: Algoritmo Genético (Genetic Algorithm)

BAB: Ramificación y Poda (Branch and Bound) es el método original , los Algoritmos BAB^{++}, $\text{BAB}^{++}_{(s)}$, RBAB y RBABM son variantes de este.

Tipo de Objetivo	Escala inicial del conjunto de características								
	Pequeño ($n < 20$)			Mediano ($20 \leq n < 50$)			Grande ($50 \leq n \leq 100$)		Muy Grande ($100 < n$)
	Mono.	Aprox. mono.	No mono.	Mono.	Aprox. Mono.	No Mono.	Mono. o Aprox. mono.	No Mono.	
A	BAB^{++}	SFFS SBFS	*	BAB^{++} $BAB^{++}_{(s)}$	SFFS SFBS GA	*	SFFS SFBS GA	*	GA
B	RBAB RBABM	RBAB RBABM	*	RBAB RBABM GA	RBAB RBABM GA	*	GA	*	GA
C	*	*	GA SFFS SBFS	*	*	GA SFFS SBFS	*	GA	GA

Tabla B.1: Recomendación de algoritmos para selección de características.

De las conclusiones de Kudo y Sklansky, se puede destacar que SFFS y SBFS dan una solución mejor que otros algoritmos de búsqueda secuencial en un tiempo razonable para los problemas de tamaño de conjunto pequeño y medianos. Los GA son adecuados para problemas de tamaño de conjuntos grandes y tienen una gran posibilidad de encontrar mejores soluciones que no pueden ser encontradas por los otros algoritmos. Los BAB y sus variaciones trabajan mejor en tamaños de conjuntos pequeños y medianos, dado que consumen mucho tiempo de ejecución.

Apéndice C

Formato Protein Data Bank

En la Figura C.1 se puede ver la sección de átomos de una estructura .pdb en la primera columna (1) se indica si es un átomo (ATOM) que forma parte de la cadena polipeptídica o en el caso contrario es un átomo que forma parte de un ligando o solo corresponde a moléculas de agua o átomos metálicos (Hierro, Cobre, Zinc, etc). La segunda columna (2) indica el número secuencial del átomo dentro del archivo. La tercera columna (3) se indica de que átomo se trata, con la nomenclatura utilizada por el PDB : N es para el Nitrógeno del grupo amida, CA es el Carbono alfa, C es el Carbono carbonilo, O es el Oxigeno carbonílico, CB es el Carbono beta, CG es el Carbono gama y por último CD1 Y CD2 son los Carbonos delta 1 y 2. La cuarta columna (4) indica el tipo de residuo en código de tres letras. La quinta Columna (5) indica a que cadena corresponde el residuo mencionado(en la columna 4). En la sexta columna (6) es el número del residuo que suele estar relacionado con la posición de este residuo en la secuencia de la proteína. Las columnas (7,8,9) se tratan de las coordenadas x,y,z del átomo en el espacio. La columna (10) es el factor de ocupancia, que quiere decir que un átomo pudiera ocupar alternativamente una de dos posiciones y cada una con probabilidad del 50 % el valor de ocupancia seria de 0.5 sino seria de 1 (probabilidad del 100 %), en general siempre es 1. La columna (11) es el factor B o factor de temperatura. La última columna (12) indica el tipo de átomo.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
416	ATOM	1	N	ALA	E	1	23.208	25.479	38.434	1.00	69.40	N
417	ATOM	2	CA	ALA	E	1	21.874	25.448	37.769	1.00	67.24	C
418	ATOM	3	C	ALA	E	1	20.763	25.477	38.815	1.00	65.79	C
419	ATOM	4	O	ALA	E	1	20.973	25.901	39.956	1.00	69.75	O
420	ATOM	5	CB	ALA	E	1	21.747	24.196	36.899	1.00	48.49	C
421	ATOM	6	N	MET	E	2	19.582	25.025	38.413	1.00	60.09	N
422	ATOM	7	CA	MET	E	2	18.423	24.980	39.293	1.00	55.22	C
423	ATOM	8	C	MET	E	2	17.880	26.343	39.721	1.00	46.03	C
424	ATOM	9	O	MET	E	2	16.735	26.637	39.421	1.00	43.58	O
425	ATOM	10	CB	MET	E	2	18.721	24.131	40.519	1.00	54.54	C
426	ATOM	11	CG	MET	E	2	17.879	22.881	40.583	1.00	58.06	C
427	ATOM	12	SD	MET	E	2	16.182	23.244	41.023	1.00	80.13	S
428	ATOM	13	CE	MET	E	2	15.297	22.720	39.523	1.00	72.12	C
429	ATOM	14	N	THR	E	3	18.663	27.168	40.414	1.00	40.24	N
430	ATOM	15	CA	THR	E	3	18.152	28.483	40.811	1.00	41.75	C
431	ATOM	16	C	THR	E	3	19.150	29.619	40.670	1.00	38.78	C
432	ATOM	17	O	THR	E	3	20.353	29.395	40.566	1.00	41.40	O
433	ATOM	18	CB	THR	E	3	17.698	28.522	42.290	1.00	31.30	C
434	ATOM	19	OG1	THR	E	3	18.835	28.317	43.127	1.00	38.11	O
435	ATOM	20	CG2	THR	E	3	16.632	27.458	42.583	1.00	32.78	C
436	ATOM	21	N	TYR	E	4	18.622	30.844	40.656	1.00	35.40	N
437	ATOM	22	CA	TYR	E	4	19.440	32.060	40.615	1.00	31.36	C
438	ATOM	23	C	TYR	E	4	18.736	33.094	41.502	1.00	33.38	C

Figura C.1: Formato Protein Data Bank sección de átomos. Obtenido usando ID: 1FS0

Apéndice D

Implementacion en Python

D.1. Algoritmos de selección de características.

Los algoritmos de selección de características implementados son Búsqueda Secuencial Hacia Adelante (SFS), Búsqueda Secuencial Hacia Atrás (SBS) y Búsqueda Flotante Secuencial Hacia Adelante (SFFS). En este Apéndice se procederá a dar explicación del código utilizado, como primera instancia se describirá un diccionario de variables continuando con las funciones que componen los algoritmos.

Variable	Tipo Variable	Descripción
subCaracteristicas	list	es una matriz que sirve en el caso del algoritmo SFS y SFFS ir almacenando las características seleccionadas y en el caso del algoritmo SBS hacer una copia de la matriz características.
caracteristicas	list	es una matriz que contiene todas las características.
cantidadFilas	int	almacena la cantidad total de filas que posee la matriz características.
cantidadColumnas	int	almacena la cantidad total de columnas que posee la matriz características.

Variable	Tipo Variable	Descripción
maximo_k	int	almacena como valor numérico la cantidad de características totales que se quieren como conjunto de salida.
maxCriterio	float	almacena la evaluación de la característica que obtenga la mayor puntuacion segun el criterio utilizado.
mejorCaracteristica	int	almacena el índice de la columna que resulto elegida como mejor caracteristica.
evaluacionCriterio	float	almacena la evaluación de las características que serán previamente comparadas.
caracteristicasFinal	list	es una matriz que es utilizada en SBS para poder ordenar las peores caracteristicas desde el final hacia el inicio.
peorCaracteristica	int	almacena el índice de la columna que resulto elegida como peor caracteristica.
archivoCompleto	file	obtiene el archivo en formato texto abriéndolo en modo lectura y obtener la matriz de caracteristicas.
matrizCaracteristicas	list	es una matriz en la cual se traspasa la matriz de características obtenida desde el archivo de texto.
cantidadPermanentes	int	almacena la cantidad de complejos permanentes existentes en la matriz de caracteristicas.
cantidadTransientes	int	almacena la cantidad de complejos transientes existentes en la matriz de caracteristicas.
indicador	list	es una lista que se agrega en la prima fila de la matriz para poder conocer la ubicacion de las caracteristicas.
k	int	sirve para poder solicitar la cantidad de características que se quieren obtener como conjunto final.
subConjunto	list	es la matriz resultante de ejecutarse algún algoritmo (SFS, SBS o SFFS).
cantidadSuboptimo	int	almacena la cantidad de características que posee la matriz subConjunto obtenida.
archivoNew	file	genera un archivo en formato texto en modo escritura para traspasar la matriz subConjunto separada por comas.

Nombre de la función	init
Entrada	-
Salida	archivo .txt del conjunto obtenido separado por comas.
Descripción	esta es la función principal, los algoritmos de selección de características tienen esta función en común solo se describirá una vez en el apéndice. La función lo que hace es traspasar las características desde el archivo de texto a una matriz en Python para poder ser utilizada, luego cuenta la cantidad de complejos permanentes y transitorios para funcionar de forma automática sin preocuparse por las dimensiones de la matriz de características. Una vez hecho esto a la matriz de características se le genera un indicador para poder conocer la ubicación de las características que son reubicadas. Teniendo la matriz de características se envía a una función de los algoritmos de selección de características (sfsMatriz, sbsMatriz o sffsMatriz). Finalmente se obtiene el resultado y se procesa para poder generar un archivo de texto con el conjunto final de caracteristicas seleccionadas.

```
1  def init():
2      #**********Preambulo**********************
3      # Obtener la matriz desde el archivo texto
4      archivoCompleto = open("prueba.txt","r")
5      lineas=archivoCompleto.readlines()
6      # Se obtiene la primera fila separada por los espacios vacios
7      # Basicamente para obtener la candidad de columnas, todas las ↩
           filas tienen el mismo largo
8      fila=lineas[0].split()
```

```
9       cantidadFilas= len(lineas)
10      cantidadColumnas= len(fila)
11
12      # Matriz de caracteristicas
13      matrizCaracteristicas=[]
14      # Inicializacion de matriz de caracteristicas
15      for i in range(cantidadFilas):
16          matrizCaracteristicas.append([0]*cantidadColumnas)
17
18      # Pasar elementos del archivo a una matriz en python
19      for f in range(cantidadFilas):
20          for c in range(cantidadColumnas):
21              matrizCaracteristicas[f][c]= float(lineas[f].split()[c])
22      archivoCompleto.close()
23
24      # Calcular la cantidad de complejos permanentes y transitorios
25      cantidadPermanentes=0
26      cantidadTransientes=0
27
28      for i in range(cantidadFilas):
29          if matrizCaracteristicas[i][0] == 1 :
30              cantidadPermanentes+=1
31          elif matrizCaracteristicas[i][0] == 2 :
32              cantidadTransientes+=1
33
34      # Agregar indicadores a la primera fila
35      cantidadCarac=cantidadColumnas
36      indicador=range(cantidadCarac)
37      matrizCaracteristicas.insert(0,indicador)
38
39      # Eliminar la primera columna perteneciente a la clase
40      matrizCaracteristicas = quitarClase(matrizCaracteristicas)
```

```
41      #************Fin Preambulo************
42
43      # El numero de caracteristicas que se quieren como resultado en ↩
            este caso todas
44      # Si se quieren menos caracteristicas cambiar el valor de k
45      k=len(matrizCaracteristicas[1])
46
47      subConjunto= sfsMatriz(matrizCaracteristicas, k, ↩
            distanciaChernoff, cantidadPermanentes, cantidadTransientes)
48      #subConjunto= sbsMatriz(matrizCaracteristicas, k, ↩
            distanciaChernoff, cantidadPermanentes, cantidadTransientes)
49      #subConjunto= sffsMatriz(matrizCaracteristicas, k, ↩
            distanciaChernoff, cantidadPermanentes, cantidadTransientes)
50
51      # Imprimir por pantalla el resultado
52      posSubObtimo=subConjunto[0]
53      cantidadSuboptimo=len(posSubObtimo)
54      for n in range(cantidadSuboptimo):
55          print "pos{}:{} ".format(n+1,posSubObtimo[n])
56
57      # Se elimina la fila de los indicadores
58      subConjunto.pop(0)
59
60      # Pasar a un archivo de texto la matriz resultante
61      archivoNew=open("pruebaRanking.txt","w")
62      for f in range(0,len(subConjunto)):
63          if f < cantidadPermanentes:
64              archivoNew.write("1")
65          else:
66              archivoNew.write("2")
67          for c in range(0,len(subConjunto[0])):
68              archivoNew.write(",")
```

```
69                archivoNew.write(str(subConjunto[f][c]))
70            archivoNew.write("\n")
71        archivoNew.close()
72
73
74    init()
```

Nombre de la función	distanciaChernoff
Entrada	recibe una sub matriz que debe ser evaluada y la cantidad de complejos permanentes y transientes.
Salida	entrega la evaluación obtenida desde la sub matriz.
Descripción	esta función representa al criterio de evaluación que se aplica a todos los algoritmo de selección de características, su explicación se encuentra en el Capitulo 5 .

```
1   def distanciaChernoff(subMatriz, n1, n2):
2       n = len(subMatriz)  #cantidad total de complejos
3
4       #probabilidad a priori
5       p1 = float(n1)/n;
6       p2 = float(n2)/n;
7
8       #matrices por clase(submatrices)
9       mPermanente=matrix(subMatriz[1:n1+1]) #se le saca la primera fila↩
            por que son los indices
10      mTransiente=matrix(subMatriz[n1+1:n])
11
12      #media transpuesta
13      M1p= mPermanente.mean(axis=0)
```

```
14      M1= M1p.T
15      M2p= mTransiente.mean(axis=0)
16      M2= M2p.T
17      #Covarianza
18      S1=np.cov(mPermanente, y=None,rowvar=0, bias=0,ddof=None)
19      S2=np.cov(mTransiente, y=None,rowvar=0, bias=0,ddof=None)
20
21      Sw1 = float(p1) * S1;
22      Sw2 = float(p2) * S2;
23      Sw  = Sw1 + Sw2;
24      probabilidad = p1*p2;
25      parte1 = float(1) / (probabilidad);
26      minimo = float(1) / 1000000; #umbral
27
28      #Valores para S1
29      S1_val,S1_vec=linalg.eigh(matrix(S1))
30      S1_val2_pos = buscarPosicionMinima(S1_val,minimo)
31      S1_vec_new  = S1_vec[S1_val2_pos, :]
32
33      #Valores para S2
34      S2_val,S2_vec=linalg.eigh(matrix(S2))
35      S2_val2_pos = buscarPosicionMinima(S2_val,minimo)
36      S2_vec_new  = S2_vec[S2_val2_pos, :]
37
38      #obtener determinante S1 elevado a la p1
39      if not S1_val2_pos: # si es vacio no existe determinante = 0
40          S1_val2_pos=range(len(S1_val)) # toma todos los valores ↩
                propios
41
42      S1_val2_new = S1_val[S1_val2_pos, :]
43      S1_fil = len(S1_val2_new)           #obtener la cantidad de filas
44                                          # determinante = 0 matriz ↩
```

```
                                                indeterminada
45
46      S1_mult = pow(S1_val2_new[0],p1)
47
48      for x in range(1,S1_fil):
49          S1_mult = S1_mult * S1_val2_new[x]
50          S1_mult= pow(S1_mult,p1)
51
52      S1_det_elevado = S1_mult
53
54      if S1_det_elevado == 0.0: #si el determinante elevado nos da cero
55          S1_det_elevado=0.1
56
57      #obtener determinante S2 elevado a la p2
58      if not S2_val2_pos: # si es vacio no existe determinante = 0
59          S2_val2_pos=range(len(S2_val)) # toma todos los valores ↩
               propios
60
61      S2_val2_new = S2_val[S2_val2_pos, :] # valores propios elegidos ↩
           mayores al minimo
62      S2_fil = len(S2_val2_new)           #obtener la cantidad de filas
63      S2_mult = pow(S2_val2_new[0],p2)
64
65      for x in range(1,S2_fil):
66          S2_mult = S2_mult * S2_val2_new[x]
67          S2_mult= pow(S2_mult,p2)
68
69      S2_det_elevado = S2_mult
70
71      if S2_det_elevado == 0.0: #si el determinante elevado nos da cero
72          S2_det_elevado= 0.1
73
```

```
74      parte2 = S1_det_elevado * S2_det_elevado
75
76      #valores de Sw
77      Sw_val,Sw_vec=linalg.eigh(matrix(Sw))
78      Sw_fil = len(Sw_val)            #obtener la cantidad de filas
79      Sw_val2_pos = buscarPosicionMinima(Sw_val,minimo)
80      Sw_vec_new =  Sw_vec[Sw_val2_pos, :]
81      Sw_val2_new = Sw_val[Sw_val2_pos, :] # valores propios nuevos
82      Sw_filn = len(Sw_val2_new)          #obtener la cantidad de filas
83
84
85      Sw_mult = Sw_val2_new[0] / parte2
86      Sw_log  = log(Sw_mult)
87      Sw_prob = parte1 * Sw_log
88
89      for k in range(1,Sw_filn):
90          Sw_log = log(Sw_val2_new[k])
91          res = parte1 * Sw_log
92          Sw_prob = Sw_prob + res
93
94      parte15 = Sw_prob
95
96      # obtener las nuevas medias desde las media M anterior, ↩
            posiciones
97      m1 = M1[Sw_val2_pos]
98      m2 = M2[Sw_val2_pos]
99      parte3  = (m1 - m2)
100
101     Sw_val_limp=[]
102
103     for i in range(Sw_fil):
104         if Sw_val[i] <= 0.0:
```

```
105                 #print Sw_val[i]
106                 Sw_val_limp.append(0.0)
107             else:
108                 #print Sw_val[i]
109                 Sw_val_limp.append(float(1)/float(Sw_val[i]))
110 
111         Sw_val_inv = diag(Sw_val_limp)
112 
113         Sw_inv  = np.dot(np.dot(Sw_vec_new,Sw_val_inv),Sw_vec_new.T)
114 
115         parte4  = np.dot(np.dot(parte3.T,Sw_inv),parte3)
116 
117         FOC= parte4 + parte15
118 
119         return  float(FOC[0][0])
```

Nombre de la función	sfsMatriz
Entrada	recibe el conjunto completo de características, el número de características que se desea, el criterio de evaluación que se utilizara y la cantidad de complejos permanentes y transientes.
Salida	entrega el subconjunto con la cantidad de características requeridas.
Descripción	comienza con la creación de una matriz **subcaracteristicas** en la cual se irán almacenando las características escogidas por el criterio de evaluación. Luego se comienza a iterar, en una iteración se comienza por escoger la característica que se encuentre en el índice 0 de la matriz **caracteristica** para luego ser comparada con las demás y obtener la **mejorCaracteristica** la cual sera agregada a la matriz **subcaracteristicas** y eliminada de la matriz **caracteristica**. Finalmente cuando se cumple con la cantidad de características requeridas se retorna la matriz **subcaracteristicas** con el conjunto ordenado según el criterio de evaluación.

```
1  def sfsMatriz(caracteristicas, maximo_k, funcionCriterio, ↩
       cantidadPermanentes, cantidadTransientes):
2
3      # Creacion de matriz del subconjunto Final
4      subCaracteristicas = []
5
6      cantidadFilas = len(caracteristicas)
7      cantidadColumnas = len(caracteristicas[0])
8      # Inicializacion de la matriz subconjunto
9      for i in range(cantidadFilas):
```

```
10          subCaracteristicas.append([]*cantidadColumnas)
11
12      # verificar cantidad de caracteristicas
13      if maximo_k > cantidadColumnas:
14          maximo_k = cantidadColumnas
15
16      pos=0 # es solo una variable para la impresion de la posicion
17
18      # Iterar hasta el maximo de caracteristicas requeridas
19      for x in range(maximo_k):
20          # se define como mejor provisorio la columna 0 de la matriz ↩
                caracteristicas
21          maxCriterio = funcionCriterio(subMatrizSFS(subCaracteristicas↩
                , caracteristicas, 0), cantidadPermanentes, ↩
                cantidadTransientes)
22          mejorCaracteristica = 0
23          for i in range(len(caracteristicas[0])-1): # comienza en la ↩
                columna 1 hasta el final para comparar
24              evaluacionCriterio = funcionCriterio(subMatrizSFS(↩
                    subCaracteristicas, caracteristicas, i+1), ↩
                    cantidadPermanentes, cantidadTransientes)
25              # se almacena en mejorCaracteristica la columna que tenga↩
                     mejor evaluacion
26              if evaluacionCriterio > maxCriterio:
27                  maxCriterio = evaluacionCriterio
28                  mejorCaracteristica = i+1
29          # Agregar columna mejor evaluada a la matriz ↩
                subCaracteristicas
30          subCaracteristicas = agregarSFS(subCaracteristicas, ↩
                caracteristicas, mejorCaracteristica)
31          print "Incluir pos: {} ; iteracion numero: {} ".format(↩
                subCaracteristicas[0][pos],x)
```

```
32          # Remover desde la matriz caracteristicas la columna mejor ↩
                evaluada
33          caracteristicas = remover(caracteristicas, ↩
                mejorCaracteristica)
34          pos+=1
35
36      return subCaracteristicas
```

Nombre de la función	sbsMatriz
Entrada	recibe el conjunto completo de características, el número de características que se desea, el criterio de evaluación que se utilizara y la cantidad de complejos permanentes y transientes.
Salida	entrega el subconjunto con la cantidad de características requeridas.
Descripción	comienza con la copia de la matriz **caracteristica** sobre la matriz **subcaracteristicas** desde la cual se irán eliminando las peores evaluaciones. Las características peor evaluadas serán guardadas en la matriz **caracteristicasFinal**, como el algoritmo SBS entrega las peores características las primeras en ser elegidas se irán ordenando desde la posición final de la matriz hasta llegar a la primera, de esta forma obtener la mejor característica en primera posición y al final la peor. Luego se comienza a iterar, en una iteración se comienza por escoger la característica que se encuentre en el la ultima posición de la matriz **subcaracteristicas** para luego ser comparada con las demás y obtener la **peorCaracteristica** la cual sera agregada a la matriz **caracteristicasFinal** y eliminada de la matriz **subCaracteristicas**. Finalmente cuando se cumple con la cantidad de características requeridas se retorna la matriz **caracteristicasFinal** con el conjunto ordenado según el criterio de evaluación.

```
1 def sbsMatriz(caracteristicas, maximo_k, funcionCriterio, ↩
      cantidadPermanentes, cantidadTransientes):
2     # Creacion de una copia en profundidad de la matriz ↩
          caracteristicas
```

```
3       subCaracteristicas= deepcopy(caracteristicas)
4       # Creacion de una matriz auxiliar para guardar las ↩
            caracteristicas
5       # Este algoritmo entrega las peores caracteristicas y se iran ↩
            almacenando desde el final hasta la primera columna
6       caracteristicasFinal=[]
7       cantidadFilas=len(caracteristicas)
8       cantidadColumnas=len(caracteristicas[0])
9       # Inicializacion de la matriz caracteristicasFinal
10      for i in range(cantidadFilas):
11          caracteristicasFinal.append([]*cantidadColumnas)
12
13      # verificar cantidad de caracteristicas
14      if maximo_k > cantidadColumnas:
15          maximo_k = cantidadColumnas
16
17      # Iterar hasta el maximo de caracteristicas requeridas
18      for x in range(maximo_k):
19
20          # se define como peor provisorio la ultima columna de la ↩
                matriz caracteristicas
21          peorCaracteristica= len(subCaracteristicas[0])-1 # elige la ↩
                ultima posicion
22          if peorCaracteristica == 0 :   # si es la ultima ↩
                caracteristica se calcula sobre ella sin eliminar y dejar ↩
                un conjunto vacio
23              maxCriterio=funcionCriterio(subCaracteristicas, ↩
                    cantidadPermanentes, cantidadTransientes)
24          else:
25              maxCriterio=funcionCriterio(subMatrizSBS(↩
                    subCaracteristicas, peorCaracteristica), ↩
                    cantidadPermanentes, cantidadTransientes)
```

```
26
27          for i in reversed(range(0,len(subCaracteristicas[0])-1)):   #↩
                comienza en la penultima columna hasta hasta la columna 0
28              evaluacionCriterio=funcionCriterio(subMatrizSBS(↩
                    subCaracteristicas, i), cantidadPermanentes, ↩
                    cantidadTransientes)
29              # se almacena en peorCaracteristica la columna que tenga ↩
                    peor evaluacion asignada
30              if evaluacionCriterio > maxCriterio:
31                  maxCriterio = evaluacionCriterio
32                  peorCaracteristica = i
33          # Agregar columna peor evaluada a la matriz ↩
                caracteristicasFinal
34          caracteristicasFinal= agregarSBS(subCaracteristicas,↩
                caracteristicasFinal,peorCaracteristica)
35          print "Excluir pos: {} ; iteracion numero: {} ".format(↩
                subCaracteristicas[0][peorCaracteristica],x)
36          # Remover desde la matriz subCaracteristicas la columna peor ↩
                evaluada
37          remover(subCaracteristicas, peorCaracteristica)
38
39      return caracteristicasFinal
```

Nombre de la función	sffsMatriz
Entrada	recibe el conjunto completo de características, el número de características que se desea, el criterio de evaluación que se utilizara y la cantidad de complejos permanentes y transientes.
Salida	entrega el subconjunto con la cantidad de características requeridas.
Descripción	comienza con la creación de una matriz **subcaracteristicas** en la cual se irán almacenando las características escogidas por el criterio de evaluación. Luego se comienza a iterar, en una iteración se comienza por escoger la característica que se encuentre en el índice 0 de la matriz **caracteristica** para luego ser comparada con las demás y obtener la **mejorCaracteristica** la cual sera agregada a la matriz **subcaracteristicas** y eliminada de la matriz **caracteristica**. Luego se procede a evaluar nuevos conjuntos que posiblemente no fueron evaluados anteriormente realizando iteraciones hacia atrás, lo que puede producir que se encuentre un subconjunto mejor al ya elegido. Para esto se ocupa el principio del algoritmo SBS. Finalmente cuando se cumple con la cantidad de características requeridas se retorna la matriz **subcaracteristicas** con el conjunto ordenado según el criterio de evaluación.

```
1  def sffsMatriz(caracteristicas, maximo_k, funcionCriterio, ↩
       cantidadPermanentes, cantidadTransientes):
2      # Creacion de matriz del subconjunto Final
3      subCaracteristicas=[]
4      cantidadFilas=len(caracteristicas)
5      cantidadColumnas=len(caracteristicas[0])
```

```
6       # Inicializacion de la matriz subconjunto
7       for i in range(cantidadFilas):
8           subCaracteristicas.append([]*cantidadColumnas)
9
10      # verificar cantidad de caracteristicas
11      if maximo_k > cantidadColumnas:
12          maximo_k = cantidadColumnas
13
14      # Iterar hasta el maximo de caracteristicas requeridas
15      for x in range(maximo_k):
16
17          if len(caracteristicas[0]) > 0:
18              # se define como mejor provisorio la columna 0 de la ↩
                    matriz caracteristicas
19              maxCriterio= funcionCriterio(subMatrizSFS(↩
                    subCaracteristicas, caracteristicas, 0), ↩
                    cantidadPermanentes, cantidadTransientes)
20              mejorCaracteristica= 0
21              if len(caracteristicas[0]) > 1:
22                  for i in range(len(caracteristicas[0])-1): # comienza↩
                        en la columna 1 hasta el final para comparar
23                      evaluacionCriterio= funcionCriterio(subMatrizSFS(↩
                            subCaracteristicas, caracteristicas, i+1), ↩
                            cantidadPermanentes, cantidadTransientes)
24                      # se almacena en mejorCaracteristica la columna ↩
                            que tenga mejor evaluacion
25                      if evaluacionCriterio > maxCriterio:
26                          maxCriterio = evaluacionCriterio
27                          mejorCaracteristica = i+1
28              # Agregar columna mejor evaluada a la matriz ↩
                    subCaracteristicas
29              subCaracteristicas = agregarSFS(subCaracteristicas, ↩
```

```
                caracteristicas, mejorCaracteristica)
30              print "Incluir pos {}; iteracion {}".format(↩
                    subCaracteristicas[0][len(subCaracteristicas[0])-1],x)
31              # Remover desde la matriz caracteristicas la columna ↩
                    mejor evaluada
32              caracteristicas = remover(caracteristicas, ↩
                    mejorCaracteristica)
33
34              peorValorCaracteristica = None # se utiliza el objeto ↩
                    especial None (Vacio)
35              if len(subCaracteristicas[0]) >= 2:
36                  for i in reversed(range(0,len(subCaracteristicas[0]))↩
                        ): # comienza en la penultima columna hasta hasta ↩
                        la columna 0
37                      # se almacena en peorCaracteristica la columna ↩
                            que tenga peor evaluacion asignada
38                      evaluacionCriterio=funcionCriterio(subMatrizSBS(↩
                            subCaracteristicas, i), cantidadPermanentes, ↩
                            cantidadTransientes)
39                      if evaluacionCriterio > maxCriterio:
40                          maxCriterio = evaluacionCriterio
41                          peorCaracteristica = i
42                          peorValorCaracteristica=peorCaracteristica
43              if peorValorCaracteristica is not None:
44                  # Remover desde la matriz subCaracteristicas la ↩
                        columna peor evaluada
45                  remover(subCaracteristicas, peorValorCaracteristica)
46                  print "Excluir pos {} ; iteracion {}".format(↩
                        peorValorCaracteristica,x)
47
48
49      return subCaracteristicas
```

Nombre de la función	agregarSFS
Entrada	recibe una sub matriz, una matriz completa y la posición que se desea agregar.
Salida	entrega la sub matriz actualizada con una nueva característica.
Descripción	esta función sirve para agregar una nueva característica a la **subMatriz**, desde una posición indicada de la **matriz** .

```
1  def agregarSFS(subMatriz,matriz,pos):
2      for i in range(len(matriz)):
3          subMatriz[i].append(matriz[i][pos])
4      return subMatriz
```

Nombre de la función	remover
Entrada	recibe una matriz y la posición.
Salida	matriz actualizada con la eliminación de la característica en la posición indicada.
Descripción	esta función cumple con eliminar desde la **matriz** una característica indicada a través de una posición.

```
1
2  def remover(matriz,pos):
3      for i in range(len(matriz)):
4          matriz[i].remove(matriz[i][pos])
5      return matriz
```

Nombre de la función	subMatrizSFS
Entrada	recibe una sub matriz, una matriz y la posición.
Salida	genera una sub matriz.
Descripción	esta función a partir de la **subMatriz** y la **matriz** genera una nueva sub matriz **aux**, esta nueva sub matriz contiene todas las características que se encuentran en **subMatriz** y agrega una nueva característica de **matriz** según la posición indicada.

```
1
2  def subMatrizSFS(subMatriz,matriz,pos):
3      aux=[]
4      cantidadFilas=len(matriz)
5      cantidadColumnas=len(subMatriz[0])
6      # Inicializacion de la matriz aux
7      for i in range(cantidadFilas):
8          aux.append([]*cantidadColumnas)
9
10     for i in range(cantidadFilas):
11         for j in range(cantidadColumnas):
12             aux[i].append(subMatriz[i][j]) # agregando ↩
                   subCaracterisiticas
13
14         aux[i].append(matriz[i][pos]) # agregando candidatas
15     return aux
```

Nombre de la función	quitarClase
Entrada	recibe una matriz.
Salida	entrega la matriz actualizada.
Descripción	esta función elimina la columna perteneciente a la clase.

```
1
2  def quitarClase(matriz):
3      # la primera columna es la clase
4      cantidadClase= len(matriz)
5      for i in range(cantidadClase):
6          matriz[i].remove(matriz[i][0])
7      return matriz
```

Nombre de la función	buscarPosicionMinima
Entrada	recibe los valores propios y el valor mínimo que pueden tener.
Salida	una lista con los valores propios que cumplen con el mínimo.
Descripción	analiza los valores propios que cumplan con el mínimo requerido, esta función es usada en la distancia de chernoff

```
1
2  def buscarPosicionMinima(valoresPropios,min):
3      aux=[]
4      for i in range(len(valoresPropios)):
5          if valoresPropios[i] > min :
6              aux.append(i)
7      return aux
```

Nombre de la función	agregarSBS
Entrada	recibe una sub matriz, una matriz y la posición.
Salida	entrega una matriz actualizada.
Descripción	esta función cumple con ingresar las características peores evaluadas al final de la **matrizFinal** ordenándolas de mejor a peor.

```
def agregarSBS(subMatriz,matrizFinal,pos):
    cantidadFinal=len(matrizFinal)
    for k in range(cantidadFinal):
        matrizFinal[k].insert(0,subMatriz[k][pos])
    return matrizFinal
```

Nombre de la función	subMatrizSBS
Entrada	recibe una sub matriz y la posición.
Salida	entrega una sub matriz.
Descripción	esta función elimina una característica de la **subMatriz** para retornar una sub matriz **aux**.

```
def subMatrizSBS(subMatriz,pos):
    aux=[]
    filas= len(subMatriz)
    columnas= len(subMatriz[0])
    for i in range(filas):
        aux.append([]*columnas)
```

```
7
8       for f in range(filas):
9           for c in range(columnas):
10              if c != pos: # saca la posicion para crear el subconjunto
11                  aux[f].append(subMatriz[f][c])
12      return aux
```

D.2. Algoritmo para generar archivo compatible con Weka.

Nombre de la función	archivoArff
Entrada	-
Salida	archivo en formato .arff
Descripción	genera un archivo .arff con el formato compatible para Weka sobre los conjuntos utilizados en esta investigación. Solo se debe configurar los nombres de los archivos en las variables archivoRead y archivoArff. Además se debe tener en cuenta que la matriz de entrada debe estar separada por comas.

Se debe escribir el siguiente código en un archivo llamado `archivoArff.py`:

```
1
2  def archivoArff():
3      archivoRead=open("pruebaRanking.txt","r")
4      archivoArff=open("pruebaRankingArff.arff","w")
5
6
7      #Cabecera
8      archivoArff.write("@relation PPI\n")
9      archivoArff.write("\n")
10     #Declaraciones de atributos
11     archivoArff.write("@attribute Clase {1,2}\n")
12
13     lineas = archivoRead.readlines()
14
15     contenido= lineas[0].split(",")
16
```

```
17      print len(contenido)
18
19      for i in range(1,len(contenido)):
20
21          if '.' in contenido[i]:
22              archivoArff.write("@attribute asd_"+str(i)+" "+"real\n" )
23          else:
24              archivoArff.write("@attribute asd_"+str(i)+" "+"numeric\n↩
                    " )
25
26      archivoArff.write("\n")
27      archivoArff.write("@data\n")
28      archivoArff.write("\n")
29      for x in range(len(lineas)):
30          archivoArff.write(lineas[x])
31
32      archivoRead.close()
33      archivoArff.close()
```

Ahora se debe compilar usando `la consola`:

```
$ python archivoArff.py
```

Apéndice E

Resultados Completos

E.1. Conjuntos de datos sin ranking

Tabla E.1: ConjuntoE(20+,20-)

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	62,8378	69,5946	64,1892	70,6081	68,2432	77,027
5-pliegues	69,2568	74,3243	71,2838	73,3108	66,8919	78,0405
8-pliegues	71,2838	75,3378	65,8784	74,3243	68,2432	79,3919
10-pliegues	68,9189	75,3378	69,5946	72,973	67,5676	76,0135
12-pliegues	69,9324	74,6622	69,5946	74,6622	67,2297	78,3784
14-pliegues	71,6216	76,0135	68,2432	71,6216	67,9054	79,0541
15-pliegues	72,6351	76,0135	69,9324	75	66,8919	78,3784
20-80	66,2447	67,0886	64,135	67,5105	67,9325	75,9494
38-62	62,5	71,7391	67,3913	67,9348	67,3913	77,1739
50-50	68,9189	75	60,8108	70,9459	68,2432	76,3514
66-34	72,2772	76,2376	71,2871	70,297	71,2871	78,2178
70-30	71,9101	78,6517	73,0337	69,6629	70,7865	78,6517
74-26	70,1299	74,026	68,8312	67,5325	71,4286	77,9221
78-22	67,6927	70,7692	69,2308	67,6923	69,2308	76,9231
80-20	67,7966	74,5763	67,7966	67,7966	67,7966	76,2712
81-19	67,8571	76,7857	69,6429	69,6429	69,6429	76,7857
90-10	70	80	70	73,3333	66,6667	83,3333

Tabla E.2: TopE(-)

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	78,1879	78,1879	75,5034	76,1745	67,1141	75,1678
5-pliegues	78,5235	78,5235	76,5101	77,1812	64,4295	78,1879
8-pliegues	79,8658	77,5168	77,5168	77,1812	63,0872	77,5168
10-pliegues	81,5436	79,1946	78,5235	77,5168	63,7584	79,5302
12-pliegues	79,5302	79,1946	78,1879	77,5168	63,0872	78,5235
14-pliegues	79,8658	79,5302	78,8591	76,5101	64,4295	77,5168
15-pliegues	80,5369	79,1946	78,5235	77,5168	64,4295	78,5235
20-80	73,1092	75,6303	69,7479	74,7899	68,4874	75,6303
38-62	75,1351	77,2973	70,2703	75,6757	65,4054	76,7568
50-50	67,7852	76,5101	71,1409	77,1812	67,7852	78,5235
74-26	77,9221	80,5195	75,3247	79,2208	74,026	76,6234
78-22	81,8182	81,8182	84,8485	78,7879	72,7273	84,8485
80-20	80	80	83,3333	76,6667	73,3333	81,6667
81-19	80,7018	80,7018	82,4561	77,193	73,6842	78,9474
84-16	77,0833	83,3333	87,5	77,0833	77,0833	83,3333
89-11	81,8182	81,8182	84,8485	78,7879	81,8182	87,8788
90-10	80	86,6667	86,6667	80	83,3333	80

Tabla E.3: TopE(+)

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	72,8188	78,1879	74,4966	75,1678	67,7852	73,4899
5-pliegues	72,8188	76,1745	75,8389	77,1812	66,443	74,1611
8-pliegues	74,8322	77,1812	73,8255	77,5168	66,1074	75,8389
10-pliegues	73,8255	76,1745	72,1477	77,1812	66,443	72,8188
12-pliegues	74,4966	77,1812	77,1812	76,8456	66,7785	72,4832
14-pliegues	73,8255	75,5034	73,8255	77,5168	67,1141	70,1342
15-pliegues	72,1477	75,1678	75,5034	76,1745	67,1141	74,1611
20-80	71,4286	79,4118	77,3109	73,1092	68,4874	71,8487
66-34	77,2277	76,2376	75,2475	71,2871	67,3267	69,3069
70-30	80,8989	77,5281	75,2809	73,0337	70,7865	74,1573
71-29	82,5581	76,7442	79,0698	70,9302	69,7674	68,6047
74-26	79,2208	75,3247	75,3247	71,4286	72,7273	71,4286
80-20	76,6667	73,3333	75	73,3333	73,3333	66,6667
81-19	78,9474	77,193	77,193	71,9298	73,6842	66,6667
86-14	83,3333	78,5714	73,8095	73,8095	78,5714	66,6667
88-12	83,3333	75	72,2222	72,2222	80,5556	66,6667
90-10	80	73,3333	70	76,6667	83,3333	76,6667

Tabla E.4: TopE(-)(+)

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	69,4631	77,8523	69,7987	75,5034	67,4497	78,5235
5-pliegues	75,5034	76,5101	73,4899	78,8591	66,443	78,5235
8-pliegues	73,4899	77,1812	72,8188	77,5168	66,7785	80,5369
10-pliegues	75,8389	76,5101	72,1477	77,5168	66,1074	81,2081
12-pliegues	73,1544	77,1812	75,5034	76,8456	66,1074	80,8725
14-pliegues	73,8255	76,1745	72,1477	77,5168	66,1074	81,2081
15-pliegues	75,1678	76,5101	74,1611	77,8523	67,1141	78,8591
20-80	73,1092	79,8319	74,3697	73,1092	68,4874	77,3109
50-50	70,4698	77,1812	67,7852	79,1946	67,7852	81,2081
66-34	73,2673	74,2574	70,297	71,2871	66,3366	77,2277
70-30	76,4045	77,5281	74,1573	73,0337	69,6629	78,6517
74-26	75,3247	74,026	74,026	70,1299	71,4286	80,5195
78-22	72,7273	77,2727	71,2121	74,2424	72,7273	83,3333
80-20	71,6667	75	68,3333	71,6667	73,3333	83,3333
81-19	73,6842	77,193	73,6842	71,9298	73,6842	84,2105
89-11	72,7273	81,8182	69,697	75,7576	81,8182	87,8788
90-10	73,3333	80	66,6667	76,6667	83,3333	83,3333

E.2. Conjuntos de datos con ranking

Tabla E.5: Conjunto(20+,20-)SFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	70,2703	69,9324	65,2027	68,5811	68,5811	77,3649
5-pliegues	72,973	70,9459	65,2027	68,2432	68,5811	79,0541
8-pliegues	71,2838	69,9324	65,5405	68,2432	68,5811	78,3784
10-pliegues	73,9865	70,2703	68,2432	68,2432	68,5811	77,7027
12-pliegues	73,6486	68,9189	63,8514	68,5811	68,5811	78,3784
14-pliegues	70,6081	69,2568	65,8784	68,2432	68,5811	76,0135
15-pliegues	73,3108	69,2568	65,2027	68,2432	68,5811	78,7162
20-80	73,8397	72,5738	69,1983	67,9325	67,9325	73,8397
38-62	75	71,7391	67,3913	67,9348	67,9348	76,6304
50-50	65,5405	66,2162	59,4595	68,9189	69,5946	75
66-34	70,297	66,3366	62,3762	72,2772	72,2772	79,2079
70-30	66,2921	65,1685	60,6742	71,9101	71,9101	78,6517
74-26	66,2338	62,3377	59,7403	71,4286	71,4286	79,2208
78-22	66,1538	61,5385	58,4615	69,2308	69,2308	76,9231
80-20	71,1864	62,7119	49,1525	67,7966	67,7966	79,661
89-11	69,697	63,6364	54,5455	66,6667	66,6667	87,8788
90-10	70	60	53,3333	66,6667	66,6667	83,3333

Tabla E.6: Conjunto(20+,20-)SBS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	65,5405	65,8784	63,8514	68,5811	68,5811	78,0405
5-pliegues	65,5405	64,527	61,1486	68,2432	68,5811	76,6892
8-pliegues	68,9189	67,5676	62,8378	68,2432	68,5811	75,6757
10-pliegues	66,5541	63,8514	60,1351	68,2432	68,5811	76,6892
12-pliegues	67,5676	63,5135	62,8378	68,5811	68,5811	77,7027
14-pliegues	66,8919	66,5541	62,8378	68,2432	68,5811	78,0405
15-pliegues	68,5811	65,8784	61,8514	68,2432	68,5811	77,3649
20-80	70,4641	71,308	65,4008	67,9325	67,9325	73,4177
38-62	67,3913	71,1957	66,3043	67,9348	67,9348	75
50-50	71,6216	73,6486	66,2162	68,9189	69,5946	77,7027
66-34	70,297	75,2475	69,3069	77,2772	72,2772	80,198
70-30	68,5393	74,1573	67,4157	71,9101	71,9101	82,0225
74-26	70,1299	76,6234	66,2338	71,4286	71,4286	77,9221
78-22	66,1538	73,8462	67,6923	69,2308	69,2308	75,3846
80-20	67,7966	77,9661	74,5763	67,7966	67,7966	77,9661
81-19	71,4286	82,1429	73,2143	69,6429	69,6429	80,3571
90-10	73,3333	76,6667	60	66,6667	66,6667	86,6667

Tabla E.7: Conjunto(20+,20-)SFFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	68,5811	70,2703	66,2162	68,5811	68,5811	72,2973
5-pliegues	68,2432	67,9054	61,4865	68,2432	68,5811	72,973
8-pliegues	66,2162	68,5811	62,8378	68,2432	68,5811	73,3108
10-pliegues	67,9054	69,2568	63,5135	68,2432	68,5811	71,9595
12-pliegues	65,5405	67,5676	62,5	68,5811	68,5811	73,318
14-pliegues	66,2162	65,8784	61,1486	68,2432	68,5811	75,3378
15-pliegues	64,1892	67,2297	61,8243	68,2432	68,5811	73,3108
20-80	68,7764	66,6667	66,6667	67,9325	67,9325	69,1983
38-62	65,7609	66,3043	61,9565	67,9348	67,9348	70,1087
50-50	71,6216	68,9189	62,1622	68,9189	69,5946	75
66-34	67,3267	67,3267	59,4059	72,2772	72,2772	75,2475
70-30	66,2921	68,5393	62,9213	71,9101	71,9101	77,5281
74-26	63,6364	66,2338	59,7403	71,4286	71,4286	77,9221
78-22	66,1538	69,2308	61,5385	69,2308	69,2308	72,3077
80-20	66,1017	66,1017	57,6271	67,7966	67,7966	71,1864
81-19	75	67,8571	60,7143	69,6429	69,6429	75
90-10	76,6667	70	76,6667	66,6667	66,6667	80

E.3. Conjuntos de datos con ranking sin residuo

Tabla E.8: ConjuntoE(20+,20-)SFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	60,1351	64,1892	64,527	66,8919	68,9189	78,0405
5-pliegues	69,9324	70,9459	66,5541	69,5946	66,8919	78,7162
8-pliegues	68,5811	71,6216	65,2027	69,2568	68,2432	78,0405
10-pliegues	68,2432	72,2973	65,8784	68,9189	67,9054	78,0405
12-pliegues	67,9054	69,9324	66,2162	67,9054	67,9054	78,7162
14-pliegues	68,2432	69,9324	65,8784	69,5946	67,9054	77,3649
15-pliegues	65,5405	69,2568	64,527	69,5946	67,5676	77,3649
20-80	72,1519	73,4177	70,0422	67,0886	67,9325	76,7932
38-62	69,5652	72,8261	66,8478	67,9348	67,9348	77,1739
50-50	72,2973	74,3243	54,7297	66,8919	68,2432	76,3514
66-34	77,2277	72,2772	65,3465	69,3069	71,2871	77,2277
70-30	74,1573	68,5393	65,1685	70,7865	70,7865	79,7753
74-26	71,4286	67,5325	63,6364	71,4286	71,4286	76,6234
78-22	66,1538	69,2308	60	70,7692	68,2308	76,9231
80-20	71,1864	67,7966	59,322	66,1017	67,7966	77,9661
81-19	73,2143	67,8571	57,1429	67,8571	69,6429	82,1429
90-10	73,3333	63,3333	60	63,3333	66,6667	80

Tabla E.9: TopE(-)SFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	76,8456	77,1812	76,1745	76,1745	67,1141	76,5101
5-pliegues	74,4966	77,5168	72,4832	77,1812	64,4295	77,1812
8-pliegues	74,4966	76,8456	74,8322	77,1812	63,0872	76,8456
10-pliegues	74,1611	76,5101	71,8121	77,5178	63,7584	78,1879
12-pliegues	74,1611	75,8389	72,4832	77,1879	63,4228	78,8591
14-pliegues	73,8255	75,5434	71,8121	77,1812	64,4295	75,8389
15-pliegues	73,8255	75,8389	73,1544	77,5168	64,4295	76,5101
20-80	71,4286	76,4706	70,5882	74,3697	68,4874	75,6303
38-62	68,6486	75,6757	72,4324	75,1351	65,4054	75,1351
50-50	69,7987	74,4966	71,1409	76,5101	67,7852	79,8658
66-34	71,2871	74,2574	71,2871	75,2475	67,3267	76,2376
74-26	76,6234	79,2208	74,026	77,9221	74,026	76,6234
78-22	78,7879	78,7879	72,7273	78,7879	72,7273	80,303
80-20	73,3333	76,6667	71,6667	76,6667	73,3333	80
81-19	73,6842	77,193	75,4386	77,193	73,6842	80,7018
88-12	77,7778	75	72,2222	77,7778	80,5556	86,1111
90-10	80	80	76,6667	80	83,3333	83,3333

Tabla E.10: TopE(+)SFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-pliegues	64,7651	74,4966	71,1409	76,8456	67,7852	74,1611
5-pliegues	68,7919	75,5034	72,1477	78,8591	66,443	75,5034
8-pliegues	66,443	77,5168	69,4631	78,5235	66,1074	74,1611
10-pliegues	69,1275	77,1812	70,4698	78,1879	66,443	74,1611
12-pliegues	65,1007	78,8591	70,4698	77,1812	67,1141	74,8322
14-pliegues	71,1409	76,8456	70,8054	79,1946	67,1141	72,4832
15-pliegues	69,4631	75,8389	70,1342	78,1879	67,1141	73,8255
20-80	72,2689	76,8608	76,8908	76,0504	68,4874	76,4706
50-50	60,4027	73,8255	69,1275	79,8658	67,7852	75,1678
66-34	60,396	72,2772	68,3168	70,297	66,3366	71,2871
70-30	69,6629	75,2809	68,5393	71,9191	69,6629	73,0337
74-26	71,4286	79,2208	68,8312	70,1299	71,4286	72,7273
78-22	68,1818	78,7879	71,2121	72,7273	72,7273	69,697
80-20	68,3333	76,6667	73,3333	70	73,3333	71,6667
81-19	71,9298	77,193	73,6842	73,6842	73,6842	68,4211
89-11	69,697	84,8485	63,6364	75,7576	81,8182	66,6667
90-10	73,3333	83,333	70	80	83,3333	76,6667

Tabla E.11: TopE(-)(+)SFS

Entrenamiento	SVM					RF
Prueba	Lineal	Polyn 2	Polyn 3	Radial	Sigmoid	
2-fold	69,1275	69,7987	69,1275	68,4564	67,1141	76,1745
5-fold	67,4497	69,1275	67,1141	68,1208	65,4362	80,5369
8-fold	70,4698	70,1342	67,4497	68,4564	64,7651	78,1879
10-fold	67,4497	67,7852	70,1342	68,1208	64,7651	79,5302
12-fold	70,1342	70,4698	68,4564	68,4564	66,1074	79,1946
14-fold	71,8121	70,1342	71,1409	68,4564	65,7718	80,2013
15-fold	66,1074	68,4564	69,7987	68,4564	67,1141	77,5168
20-80	70,5882	69,3277	68,4874	69,7479	68,4874	74,7899
38-62	67,5676	68,1081	69,1892	68,1081	65,4054	73,5135
50-50	70,4698	69,1275	66,443	69,7987	67,7852	79,1946
66-34	62,3762	64,3564	64,3564	69,3069	67,3267	73,2673
70-30	62,9213	68,5393	66,2921	71,9101	70,7865	77,5281
74-26	63,6364	68,8312	64,9351	74,026	72,7273	75,3247
78-22	66,6667	66,6667	68,1818	74,2424	72,7273	78,7879
80-20	70	70	66,6667	71,6667	73,3333	78,3333
81-19	70,1754	68,4211	64,9123	71,9298	73,6842	80,7818
90-10	60	60	63,3333	80	83,3333	86,6667