UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

MEJORAMIENTO DE LA CLASIFICACIÓN FUNCIONAL DE ENZIMAS USANDO APRENDIZAJE DE MÁQUINAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DAVID IGNACIO GÓMEZ PADILLA

PROFESOR GUÍA:
RICARDO NILO POYANCO

MIEMBROS DE LA COMISIÓN:
FELIPE TOBAR HENRÍQUEZ
ANDRÉS CABA RUTTE

Este trabajo ha sido parcialmente financiado por CONICYT 11150107

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: DAVID IGNACIO GÓMEZ PADILLA
FECHA: 2017
PROF. GUÍA: SR. RICARDO NILO POYANCO

MEJORAMIENTO DE LA CLASIFICACIÓN FUNCIONAL DE ENZIMAS USANDO APRENDIZAJE DE MÁQUINAS

Los avances tecnológicos han permitido secuenciar el ADN de un organismo de manera mucho más accesible que en el pasado. Esto ha generado grandes volúmenes de información; en particular, de los principales productos génicos, las proteínas. Sin embargo, solo se ha logrado asignar funcionalidad a una centésima parte de las proteínas disponibles, ya que ello se realiza de forma experimental, lo cual es muy laborioso y lento. Es por ello que se han desarrollado un gran número de métodos computacionales que buscan predecir la funcionalidad de las proteínas. Dentro de ellos, BLAST (*Basic Local Alignment Search Tool*) ha sido el más usado, el cual asigna funcionalidad basándose en la noción de homología: proteínas con secuencias aminoacídicas similares tendrían funciones similares. Sin embargo se ha visto que proteínas con secuencias muy distintas pueden tener la misma funcionalidad, y variaciones en la secuencia de una proteína pueden tener grandes impactos en su función. Debido a las limitaciones de la inferencia de funcionalidad basado en homología, numerosos acercamientos basados en aprendizaje de máquinas han sido propuestos como alternativas. CAFA (*Critical Assesment of Functional Annotation*) es una competencia que busca evaluar las distintas alternativas que han surgido. Este desafío ha arrojado que no existe un método que sobrepase claramente a los demás, además de probar que si bien las alternativas propuestas sobrepasan el rendimiento de BLAST, este último aún sigue teniendo efectividad.

En el presente trabajo se propone BLAST-KNN: un algoritmo que ensambla técnicas de aprendizaje de máquinas junto a BLAST para mejorar el proceso de clasificación funcional en enzimas, un subconjunto de las proteínas, utilizando la nomenclatura de los números EC (*Enzyme Commission*) como etiquetas. De esta manera se aprovecha la efectividad de BLAST y se intentan corregir aquellas clases en que este no tiene un rendimiento perfecto. Se incorpora el uso del programa InterProScan como extractor de características para representar las proteínas, lo que entrega la ventaja de tener información basada no solo en homología. Se seleccionan las características más relevantes usando técnicas de teoría de la información. Usando los datos disponible en SwissProt que cuentan con sus cuatro dígitos EC asignados, se logran mejorar 835 clases en términos del puntaje F1 obtenido solo por BLAST, lo que representa el 55.48 % de las clases en que BLAST no tiene un rendimiento perfecto. Además, se muestra un predominio de BLAST-KNN frente a BLAST al evaluar clases con más de un número EC asignado, mejorando el 60.3 % de los casos. Por otro lado, se valida PANTHER, CDD y los descriptores propios de InterPro (IPR) como fuente importante de información al momento de predecir números EC a nuevas enzimas.

Las limitantes del algoritmo propuesto están en la poca información por clase disponible, teniendo una distribución no uniforme en el número de muestras por etiquetas, lo que fuerza a utilizar algoritmos simples para clasificar. Se propone mejorar la representación de las enzimas incorporando nuevas características, así como extender el clasificador a uno que considere enzimas que no tengan los cuatro dígitos EC asignados.

Agradecimientos

A mis profesores Felipe Tobar y Ricardo Nilo, a los cuales debo agradecer la dedicación y el compromiso que pusieron para que esto saliera adelante. Sinceramente, no he escuchado de ningún otro profesor que reciba mail's a las 00:00 AM y los responda a las 2 AM. Gracias Ricardo por eso, por la confianza y las largas discusiones que dieron pie a este trabajo, apoyados por Guillermo.

Ni las mejores amistades te conocen tanto como lo hace tu propia familia. Aquellos que con solo saludar al llegar a casa saben como fue el día. Gracias por apoyar y soportar toda esta etapa final, y no solo eso, sino todos estos años de crecimiento y mantener la fe intacta en mí. Ya tenemos ganadora del concurso *¿En qué mes me titulo?*. Sin duda las idas a nuestro querido Pichilemu fueron un impulso para terminar todo esto. Eso si lo vale.

La deuda académica que dejó la U es compensada por la gente que trajo. Migue, un reencuentro luego de compartir desde la básica, sin duda el partido que jugaremos nunca será solo. Feña, ya eres una más de *Los Bodrios*, gracias por tanto cariño que trajiste en forma de cupcakes. Rodrigo Muñoz, *loquillo*, aprendí más de ti que de cualquier profesor, fuiste un mentor y gracias a eso hoy trabajo en lo que me gusta. Te faltó aprender de mi en *Counter*. Anto, todos esos códigos debugueados, presentaciones a última hora e intentos de ser eléctricos me dejan a la mejor compañera de trabajo y amiga. Si mi futuro se aclaró, también lo hará el tuyo. Catan y papas fritas pendientes!.
A todos los electroviejones, por las conversaciones, asados y apoyo mutuo en estos años. Siempre será viernes cuando nos reunamos. A la natación y nuestro reencuentro universitario, fundamental hasta el día de hoy. A la Rama de Natación de Ingeniería (gigantes *papito* y *zorrito*) y a mi querida piscina de Cal y Canto, de profesores (Moni, siempre dándome la confianza en los torneos) a compañeros (en especial la gente de *Salud*).

Los de siempre no necesitan mayores agradecimientos. Incondicionales. A *Los Bodrios* Xapu, Manche, Papa y Chile, ya van 12 años. Chile, el colegio no se supera, se adapta. Adrián, *Mi General*, lo más grande, el distinto, sabemos de donde proviene tu potencial. Mila, *matejusticiera*, la distancia física no le hace peso a nuestra etapa de polluelos esperando las 4 micros. No nos perdamos. Mamey, *cóndor*, barrio, colegio, carrera, viajes. De la banca en la plaza al Esperanto. El tiempo pasa pero somos la constante. Tu consejo me formó, pero sabes bien que el aprendiz superó al maestro. Gracias por tanto.
Y a ti, que hiciste parte de ti todo lo que fui pasando en esta etapa. Ahora solo queda reír y crecer. El sur nos iluminó, no lo rompamos.

Se acabaron las holguras

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

La genómica se entiende como el área dedicada al estudio de los genomas [1]. Estos últimos corresponden al conjunto de genes que almacenan toda la información de un organismo o especie. Es una de las áreas más vanguardistas de la biología, puesto que el impacto que tiene ha permitido avances tanto en medicina como en agricultura. Famoso fue el Proyecto Genoma Humano en los 90's, cuyo objetivo era identificar y mapear los genes del genoma humano [2]. El uso de la genómica permitiría conocer la base molecular de muchas enfermedades con lo cual se puede realizar un mejor diagnóstico. Para realizar esto último, el siguiente paso en el estudio de un sistema biológico es la proteómica. En ella se estudia a gran escala las proteínas, su estructura y función, usando la enorme cantidad de información obtenida de los genes de diversos organismos. El estudio de las proteínas ante diferentes condiciones permitiría identificar aquellas proteínas cuya presencia o ausencia están correlacionadas con determinados estados fisiológicos.

En los últimos años, ha habido un importante auge en estas áreas gracias a los avances tecnológicos que permiten secuenciar el ADN de un organismo de manera mucho más accesible que en el pasado. Si el Proyecto Genoma Humano fue dotado inicialmente de 3000 millones de dólares, hoy en día se puede secuenciar un genoma humano por no más de 1000 dólares [2]. Esto ha generado grandes volúmenes de información disponible para analizar; en la fig 1.1 [3] se muestra como ejemplo el aumento de información disponible en un subconjunto de una de las bases de datos de proteínas más reconocidas a nivel mundial (SwissProt de UnitProt [5]). Si la limitante en el pasado era la falta de información, hoy en día es la poca capacidad de procesar ese gran volumen de datos, en parte porquelos experimentos para determinar manualmente la funcionalidad de una proteína son costosos y lentos. En la tabla 1.1 se muestra la distribución de proteínas que hay en UnitProt, considerando las que han sido manualmente revisadas/anotadas (Swiss-Prot) y las que han sido anotadas mediante el uso de métodos computacionales (TrEMBL). Es por ello que un gran número de métodos computacionales que puedan predecir funcionalidad han sido

[1] Información extraída de https://ghr.nlm.nih.gov/primer/hgp/genome

[2] Información extraída de https://www.genome.gov/sequencingcosts/

[3] www.unitprot.org

desarrollados en los últimos años como una alternativa de bajo costo a la anotación basada en métodos experimentales o la curación manual. Estos se basan en información de la secuencia, estructura o funcionalidad de la proteína.

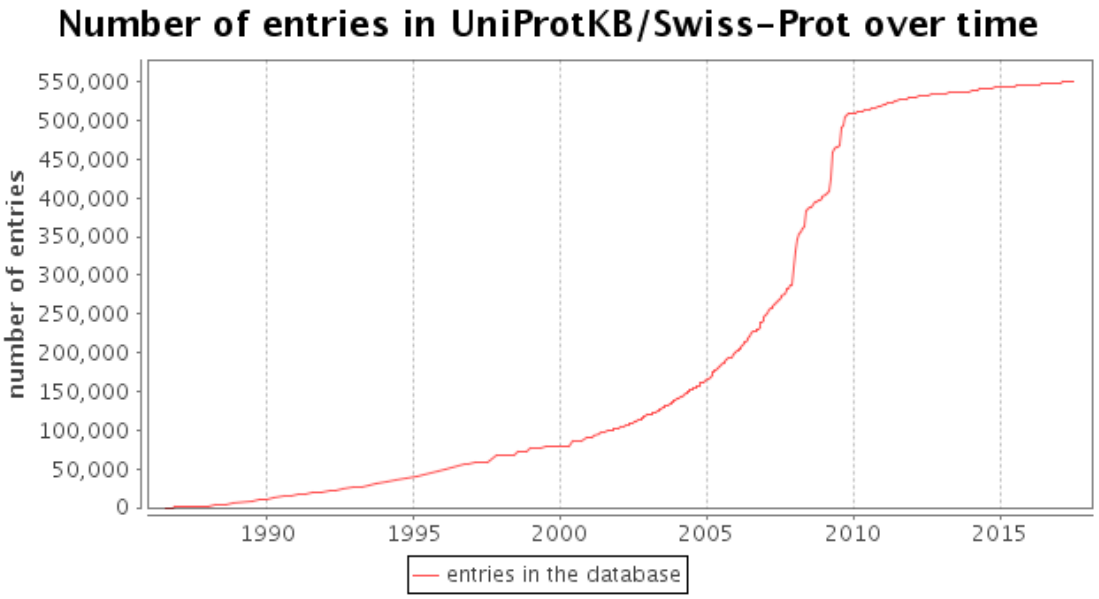


Figura 1.1: Evolución del número de entradas de base de datos SwissProt. Imagen extraída de [3].

Base de datos	Número de proteínas
Swiss-Prot	555.100
TrEMBL	88.032.926

Tabla 1.1: Distribución de proteínas en base de datos UnitProt *release* 2017_07. Datos obtenidos de [3].

Los métodos basados en alineación de secuencias han sido la manera más común de asignar funcionalidad a nuevas proteínas. Se basan en el concepto de homología: si dos proteínas tienen secuencias similares es porque provienen de un ancestro común y, al ser homólogas, tendrían una estructura tridimensional similar y frecuentemente esto indica similitud en la función que realizan [6]. BLAST [7] ha sido el programa más usado para transferir funcionalidad por similitud. Sin embargo, existen contra-ejemplos que muestran que pequeños cambios locales en la secuencia pueden llevar a drásticas variaciones en la funcionalidad, así como ejemplos de proteínas con baja homología y funciones similares cuyas secuencias han divergido dada su gran distancia evolucionaria en el tiempo, pero que provienen del mismo ancestro. Todas estas excepciones provienen del hecho de que la similitud estructural entre proteínas es lo que normalmente define la función de estas. Sin embargo, presentar una secuencia aminoacídica similar no implica que las estructuras tridimensionales lo van a ser.

Existen diversas maneras de organizar la información en base a funcionalidad. Para el caso particular de las proteínas que catalizan reacciones bioquímicas (enzimas), el estándar clásico corresponde a los números EC, los cuales corresponden a una manera jerárquica de clasificación, con cuatro dígitos en que

cada uno va agregando información acerca de la funcionalidad, desde lo más general (primer dígito) hasta la especificidad (último dígito) [8].

Debido a las limitaciones de la asignación de funcionalidad por homología, numerosos acercamientos se han propuesto. PSI-BLAST [7] o HMMER [9] utilizan la alineación de múltiples secuencias simultáneamente para extraer regiones conservadas en familias de proteínas y así poder identificar homólogos lejanos. Sin embargo se siguen basando en homología. La predicción de funcionalidad usando estructura tridimensional se ha visto muy limitada por la baja cantidad de información estructural que se ha generado hasta el momento. El uso de la secuencia es por lejos lo más usado debido a la facilidad de obtenerla y para ello se han empleado numerosos algoritmos basados en *machine learning* (aprendizaje de máquinas), los cuáles se basan en el uso de datos de entrenamiento para descubrir patrones ocultos, construir modelos y hacer predicciones. Su capacidad de predicción radica mayormente en la manera de representar los datos; las llamadas características. Existe un amplia variedad de trabajos que experimentan con diversas maneras de representar la secuencia de cada proteína: propiedades biológicas [10], dominios [11], uso de interacción entre proteínas (PPI) [12], herramientas de procesamiento natural de lenguaje como *word2vec* [13], así como con diversos algoritmos tales como *Support Vectors Machine* [14], Random Forest [15], *K-Nearest Neighbors* o el uso de redes neuronales [16]. Si bien la mayoría muestra resultados promisorios, varios de ellos se limitan a ciertos subconjuntos del total de datos, asumiendo condiciones ideales (clases con gran volumen de datos o una clasificación con baja especificidad) y no funcionan adecuadamente cuando se ven enfrentados a clasificaciones que buscan mayor nivel de detalle. CAFA (*Critical Assessment of Functional Annotation* [3]) es un desafío que busca evaluar los métodos existentes usando un *benchmark* para así poder compararlos entr ellos y al acercamiento tradicional de asignación de funcionalidad por similitud (BLAST). Con esto se tiene una idea de los avances y los caminos a seguir. Las principales conclusiones del último desafío CAFA muestran que los acercamientos por similitud siguen teniendo un buen rendimiento basados en su simplicidad, pero que para lograr un mejor resultado en la clasificación se requiere incorporar variadas fuentes de información y utilizar métodos ensamblados para mayor diversidad en la predicción. En el presente no existe un método que sobrepase claramente a los demás propuestos.

1.2. Objetivos

El objetivo general del presente trabajo es desarrollar un algoritmo de ensamble basado en técnicas de *machine learning* y de homología usadas actualmente, para mejorar el proceso de clasificación funcional en los casos en que los acercamientos tradicionales por homología han fallado.

Los objetivos específicos son los siguientes :

- Llevar el problema de clasificación al caso particular de las enzimas, usando como identificadores de clases la nomenclatura EC.
- Estudiar el comportamiento de BLAST sobre el conjunto de enzimas, para analizar sus ventajas y defectos como clasificador.
- Revisar el estado del arte de las técnicas de *machine learning* que se han utilizado para intentar

complementar las falencias de la búsqueda por homología.

- Definir el problema a resolver junto a sus limitaciones. Seleccionar una base de datos y en base a ella probar y validar el acercamiento propuesto
- Comparar los resultados obtenidos con el programa más utilizado al día de hoy. Analizar ventajas y desventajas
- Proponer mejoras al trabajo al igual que recomendaciones para su correcto uso en el futuro

1.3. Estructura del trabajo

En el capítulo 1 se detalla el contexto que genera la necesidad de realizar este trabajo. Se muestran los objetivos propuestos y las contribuciones realizadas. En el capítulo 2 se definen conceptos utilizados, se revisa el estado del arte en el área junto a un marco teórico de las técnicas utilizadas hoy en día y las que se utilizan en el acercamiento propuesto. En el capítulo 3 se detallan las herramientas utilizadas para la implementación del algoritmo propuesto. En el capítulo 4 se muestran las pruebas de conceptos realizadas a pequeños conjuntos de datos con el fin de probar acercamientos y maneras de representar las enzimas, para luego generar una propuesta de algoritmo a implementar. Se detalla la implementación, los pasos que lo componen y como se ensambla a BLAST. En los capítulos 5 y 6 se dan a conocer los resultados arrojados por el algoritmo ensamble propuesto y la comparación frente al uso de BLAST. Luego, se da paso una discusión en base a lo obtenido. Finalmente, en en el capítulo 7 se presentan las conclusiones de este trabajo y recomendaciones para un trabajo futuro.

Capítulo 2

Marco Teórico

2.1. Proteínas y Enzimas

Las proteínas son macromoléculas responsables de los procesos biológicos dentro de una célula. Consisten en una cadena de aminoácidos unidas mediante enlaces peptídicos. Los aminoácidos, por lo tanto, corresponden a la unidad química que lo conforman, existiendo 20 tipos distintos (tabla 2.1). Dependiendo de la secuencia aminoacídica y la interacción con su entorno, las proteínas se pliegan tridimensionalmente, permitiéndoles interactuar entre ellas y realizar distintas funciones (fig. 2.1 [1]).

Un tipo especial de proteína son las enzimas. Corresponden a las moléculas más esenciales en un organismo. Son responsables de catalizar reacciones bioquímicas que, de lo contrario, no podrían ser llevadas a cabo o que se llevarían a cabo en una cantidad de tiempo órdenes de magnitud mayores a las logradas por las enzimas. Conocer la función que realizan es uno de los mayores desafío actuales en la bioinformática, puesto que esa información es crucial para lograr reconstruir las sucesiones de reacciones químicas que ocurren dentro de un organismo; es decir, las rutas metabólicas, y con ello poder entender el cómo los organismos se mantienen internamente estables y a la vez responden a estímulos externos.

2.1.1. Estructura primaria, secundaria y terciaria

Corresponde a la manera más básica de organización de una proteína. Está determinado por su cadena de aminoácidos. Es la información más fácil de obtener y la que más abunda, a diferencia de la estructura secundaria y terciaria que se relacionan al plegamiento tridimensional que tiene esta. La desventaja de la secuencia primaria es que la función de una proteína usualmente queda determinada por la manera en que se pliega, lo cual no se puede obtener a partir de la estructura primaria. Esta limitante es la que dificulta la clasificación funcional.

[1] Imagen extraída de www.ebi.ac.uk

La manera más común de trabajar esta información es mediante el formato *fasta*. Este es un formato de fichero de texto que se utiliza para representar secuencias biológicas. Su estructura se compone de dos partes :

1. Una descripción de la secuencia luego del símbolo '>'. Se puede colocar toda la cantidad de información que se estime conveniente.
2. Bajo la línea de descripción, se ubica la secuencia.

Esto se ejemplifica en la fig. 2.2 [2].

Aminoácido	Codificación en tres letras	Codificación en una letra
Alanina	Ala	A
Cysteina	Cys	C
Ac. Aspártico	Asp	D
Ac. Glutámico	Glu	E
Fenilalanina	Phe	F
Glicina	Gly	G
Histidina	His	H
Isoleucina	Ile	I
Lisina	Lys	K
Leucina	Leu	L
Metionina	Met	M
Asparagina	Asn	N
Prolina	Pro	P
Glutamina	Gln	Q
Arginina	Arg	R
Serina	Ser	S
Treonina	Thr	T
Valina	Val	V
Triptofano	Trp	W
Tirosina	Tyr	Y

Tabla 2.1: Aminoácidos existentes y su codificación.

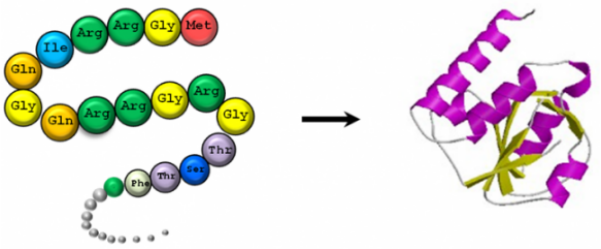


Figura 2.1: Cadena de aminoácidos y su plegamiento tridimensional. Imagen extraída de [1].

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Figura 2.2: Ejemplo de una proteína representada en el formato *fasta*. Imagen extraída de [2]

[2] www.wikipedia.org

2.1.2. Dominios

No toda la cadena aminoacídica tiene la misma relevancia. Dentro de ella existen zonas con mayor densidad de plegamientos si se observa de manera tridimensional. Estas zonas son llamados dominios. Estos pueden ser funcionales si es una unidad de la proteína que lleva a cabo una función, o estructural si es una componente estable de su estructura. Si bien son responsables de alguna función en particular, los dominios pueden existir en distintos contextos, en donde dominios similares pueden ser encontrados en proteínas con distinta funcionalidad. Asimismo, pueden encontrarse más de un dominio en una cadena, lo que podría revelar más de una funcionalidad en ciertos casos. En la fig. 2.3 [2] se muestra el plegamiento de una proteína que consta de 3 dominios.

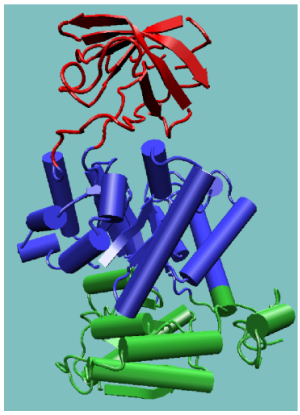


Figura 2.3: Ejemplo de plegamiento de una proteína cualquiera con tres dominios (destacados en rojo, azul y verde). Imagen extraída de [2].

2.1.3. Números EC

El estándar actual para la anotación funcional de enzimas es la nomenclatura de los números EC (*Enzyme Commission numbers*) [17]. Este conjunto de reglas dispuesto por la NC-IUBMB genera una clasificación basado en la reacción química que cada enzima cataliza. Consta de 4 dígitos en que cada nuevo dígito va agregando progresivamente más especificidad; sin embargo, es el último dígito el que entrega mayor especificidad, siendo los tres primeros aún bastante generales. Se representan como EC X.Y.W.Z,en donde X,Y,W,Z corresponden a los cuatro niveles posibles de especificidad. Cada nivel comprende una cantidad variable de opciones: para el primer dígito existen 6 alternativas (de 1 a 6), luego cada caso se va ramificando. En la fig. 2.4a [3] se muestra la cantidad de clases que existen por cada nivel.

La cantidad de enzimas clasificadas/asociadas a cada una de clases no es uniforme. La fig. 2.4b [11] muestra la distribución por cantidad de números EC ante distintas bases de datos (SwissProt, TrEMBL, KEGG) y ante distintos organismos en escala logarítmica al año 2012. En ella se muestra que un 10 % de los números EC existentes conglomera el 80 % del total de enzimas disponibles. Esto evidencia la existencia de acotadas clases muy bien representadas. El restante 90 % de clases solo contiene el 20 % del total de enzimas, lo que muestra que la mayoría de las clases EC contiene muy pocos integrantes.

[3] www.enzyme-database.org/contents.php

Como ejemplo de la clasificación EC, se muestra el caso de la enzima con entrada D5MP61 en la base de datos SwissProt. Esta contiene sus cuatro dígitos EC identificados.

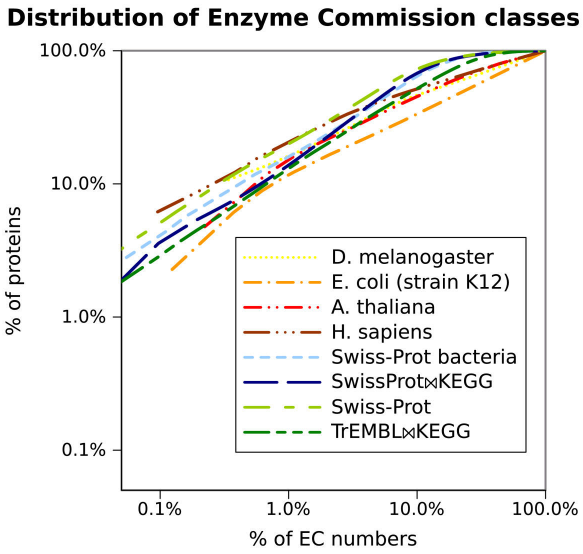
$$D5MP61 \rightarrow 3:x:x:x \rightarrow 3:2:x:x \rightarrow 3:2:1:x \rightarrow 3:2:1:32 \tag{2.1}$$

Cada enzima puede estar asociada a más de un número EC, en el caso en que participe en más de una reacción química.

Enzyme classes of the NC-IUBMB enzyme list

Class	Name	N.N.x.x	N.N.N.x	N.N.N.N
1	Oxidoreductases	26	141	1711
2	Transferases	10	37	1757
3	Hydrolases	13	66	1319
4	Lyases	8	17	618
5	Isomerases	6	17	273
6	Ligases	6	10	189
				5867

(a) Distribución de cantidad de números EC



(b) Distribución de muestras por clases EC. Imagen extraída de [11]

Figura 2.4: Distribución de números EC; por clase 2.4a y por muestras 2.4b.

2.1.4. UnitProtKB

UnitProt Knowledge Base [5] es una base de datos de información funcional de proteínas, la cual entrega un identificador, secuencia, nombre y número EC de estas, entre otras alternativas. Está dividida en dos secciones : SwissProt, la cual contiene proteínas que han sido manualmente anotadas; es decir, que la información que se entrega ha sido revisada por un conjunto de biólogos expertos, y TrEMBL, la cual contiene una cantidad mayor de proteínas que aún no han sido revisadas. La distribución de proteínas en cada una de las bases de UnitProt se mostró en el capítulo 1 (tabla 1.1). Dada la vasta diferencia en términos de cantidad de enzimas existentes entre los datos de SwissProt y TrEMBL, UnitProt contiene procesos de anotación automática que incluye árboles de decisión y conjuntos de reglas desarrollado por expertos para generar una clasificación funcional de las proteínas, utilizando información de distintas bases de datos que agrupan proteínas de acuerdo a algún criterio establecido. Con ello se intenta entregar al usuario información que, si bien no ha sido comprobada, proporciona una noción de la función de cada proteína existente.

Si bien existen otras bases de datos como BRENDA, UnitProt y en particular SwissProt son las fuentes más usadas cuando se quiere realizar algún tipo de predicción, puesto que se basan en información confiable, además de actualizarse mensualmente.

2.2. Búsqueda por Similitud: BLAST

En [7] se presenta *BLAST : Basic Local Alignment Search Tool*. Este algoritmo es por lejos el más popular en el área de bioinformática para realizar búsquedas basadas en homología de secuencias. Esto se refiere a que ante una determinada secuencia de consulta por parte del usuario, se busca en una determinada base de datos (SwissProt o TrEMBL, por ejemplo) la secuencia que obtenga el mejor puntaje de alineación. Este proceso se divide en los siguientes pasos :

- Filtrado: Se filtran las zonas de baja complejidad en la secuencia de consulta. Esto es debido a que zonas con subsecuencias repetidas no aportan mayor información a la hora de hacer las comparaciones. Para ello se utilizan máscaras sobre la secuencia de consulta, que cambian las zonas filtradas por X's o N's.
  Se define una *palabra* como una subsecuencia de largo 3 para aminoácidos por defecto. Sin embargo, el usuario puede definir el largo deseado. Por ejemplo, AGPVA produce 3 palabras de largo 3 : AGP, GPV y PVA. Por cada palabra generada, se busca en la base de datos alguna secuencia que la contenga o en su defecto, que contenga una palabra similar. El concepto de similitud viene dado por un puntaje asociado al comparar dos palabras y un umbral minino que se debe cumplir. Para determinar aquel puntaje se utiliza alguna de las matrices disponibles (BLOSUM 62 por defecto para aminoácidos) que indica la probabilidad de que dos aminoácidos sean sustitutos el uno del otro. Estas matrices fueron introducidas por primera vez en [18]. Con ella, se compara cada aminoácido correspondiente asignando un puntaje de similitud. Este primer paso se ve en la fig. 2.5 (de [4]), en donde la palabra PQG es la escogida y abajo se indican todas las palabras similares que se encontraron en alguna secuencia de la base de datos cuyo puntaje supera el umbral T.

- Extensión: Cada palabra similar es usada para alinear la secuencia a la que pertenece con la secuencia consultada. Una vez se produce la alineación, se compara aminoácido por aminoácido entre ambas secuencias, extendiendo las secuencias en ambas direcciones. Esto se realiza hasta que el puntaje global no decaiga de un valor X. En la fig. 2.5 se ejemplifica con las secuencias *query* y *subject*. Una vez se alinean las palabras similares (PQG y PMG), se comparan los aminoácidos a ambos lados usando la misma matriz de puntajes. A este proceso se le llama *High-scoring segment pair* HSP.

- Evaluación: Se calcula el puntaje total de cada secuencia alineada, obteniendo la probabilidad de que esta alineación haya sido al azar de acuerdo al tamaño de la base de datos. Se reportan los alineamientos que obtengan una probabilidad menor a un valor E. Este valor es conocido como e-value e indica el número de alineaciones esperadas al azar con un puntaje igual o mejor al obtenido. Esta métrica es más significativa que revisar la similitud de dos secuencias puesto que se han visto casos de secuencias con un porcentaje bajo de identidad (menos del 30 %) que tienen un e-value significativo ($<10^{-6}$) [6]. Por defecto se utiliza un valor de 10, por lo que se debe realizar un aná-

[4]https://www.ncbi.nlm.nih.gov/Class/MLACourse/Modules/BLAST/how_blastp_works.html

lisis posterior de los resultados para considerar si la alineación es significativa o no, dado que un valor mayor a 1 indica que podría existir algún otro alineamiento con el mismo puntaje al reportado.

Una vez que se encuentran las mejores secuencias alineadas, se puede inferir la funcionalidad de la secuencia consultada por medio de homología. Esto dice relación con asumir que secuencias similares tendrán funcionalidades similares. Se puede asignar la funcionalidad de la mejor alineación o considerar las k primeras y ver la que más se repite. Esto ya queda en manos del usuario, pues BLAST no está diseñado para inferir funcionalidad, solo para alinear secuencias.

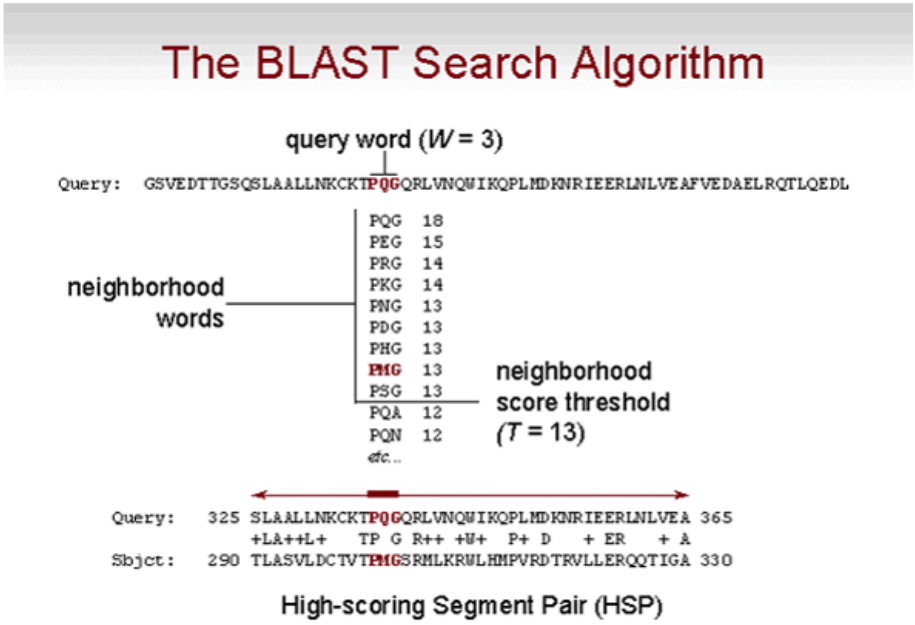


Figura 2.5: BLAST identificando una secuencia. Imagen extraída de [4].

2.3. Acercamiento a Aprendizaje de Máquinas

El campo del aprendizaje de máquinas busca desarrollar algoritmos que mejoren con la experiencia, para así asistir a humanos en el análisis de grandes y complejos conjuntos de datos. Este acercamiento ha sido aplicado a diversas áreas como negocios, robótica, meteorología, astronomía y genómica.

Los algoritmos de aprendizaje son, básicamente, modelos que se pueden ajustar en base a conjuntos de datos de entrenamiento para luego poder generalizarlos. Dentro de ello se puede encontrar entrenamientos supervisados, en donde se entregan datos con una etiqueta final que le indica a la máquina el conjunto al que pertenecen los valores ingresados. Así, la máquina busca ajustar sus parámetros para que la mayor cantidad de datos sean clasificados con la misma etiqueta que se provee. También se pueden entrenar de manera semisupervisada o no supervisada, en donde los datos se entregan sin una etiqueta que diga a que conjunto pertenecen y el mismo programa busca generar agrupamientos (*clusters*) en torno a ellos. Un ejemplo de esto se ve en la fig. 2.6, en donde se muestra un conjunto de datos de entrenamiento

de secuencias de ADN con una etiqueta que indica si un área específica del código está centrada en el punto de partida de la transcripción (TSS) o no. Mediante este entrenamiento (supervisado) realizado sobre algoritmos de aprendizaje de máquinas (redes neuronales, *Support vector machines*, clasificación bayesiana, por nombrar algunos) se genera un modelo específico para este problema, con los parámetros ajustados al caso estudiado. Luego, se presenta un conjunto de datos de validación que también representan secuencias de ADN, pero el modelo es el que debe determinar su etiqueta (TSS o Not TSS). Este representa un tipo de problema en genómica [1].

Por otro lado, estos modelos se diferencian entre generativos o discriminativos, en base a como buscan separar y clasificar los datos entregados. Se dice que un modelo es generativo cuando este genera relaciones entre los datos con una misma etiqueta, logrando así crear una fórmula que represente su distribución, con la posibilidad de generalizar. Un ejemplo son los algoritmos de *cluster*. En cambio, un modelo discriminativo no busca generar fórmulas que generen dos conjuntos separados de datos, sino que solo busca un plano que pueda separar datos con diferente etiqueta, independiente de la distribución que tenga cada conjunto. Esto se ejemplifica en la fig. 2.7.

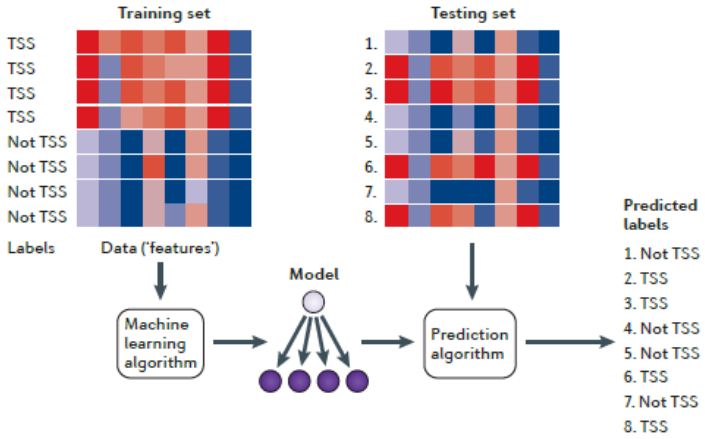


Figura 2.6: Ejemplo de aplicación de aprendizaje de máquinas en genómica. Imagen extraída de [1].

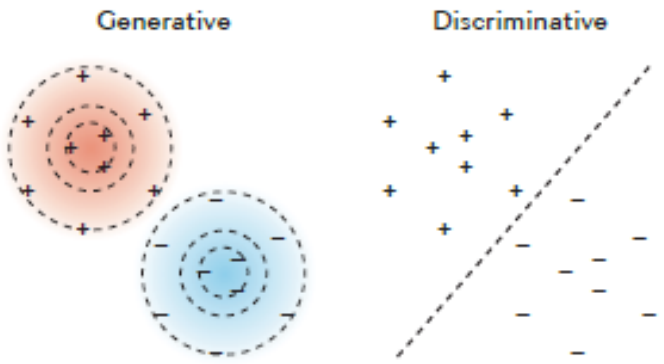


Figura 2.7: Diferencia entre modelo generativo y discriminativo. Imagen extraída de [1].

2.3.1. Extracción de características

La mayoría de los algoritmos de aprendizaje de máquinas usan como entradas vectores de largo fijo. Esto conlleva un problema si se quiere utilizar directamente las secuencias de aminoácidos, puesto que las cadenas varían de 2 a más de 4000 aminoácidos (fig. 2.8 [5]). Recortar las cadenas y dejar un largo fijo significa sesgar los datos, además de requerir un conocimiento experto avanzado para conocer dónde y qué partes cortar, por lo que la alternativa es representar cada secuencia de aminoácidos mediante un vector de características de largo fijo. La selección y extracción de características ayuda a reducir la dimensionalidad del problema, mejorar el rendimiento del algoritmo y disminuir la complejidad y los cálculos en el problema.

Algunas de las características que se pueden extraer de la secuencia aminoacídica, son:

- Composicionales : Se expresan en forma de frecuencia de aparición de aminoácidos en las cadenas.
- Composición pseudo aminoacídica : Incorpora la información del orden de la secuencia, mediante más de 20 factores, en donde los primero 20 son composicionales y el resto incluye factores de correlación, los cuales se extraen de las propiedades físico-químicas de pares de aminoácidos del tipo (i,i+1), (i, i+2) y (i,i+3).
- Dominios: Sus presencias sirven de diagnóstico para predecir cierta funcionalidad o pertenencia a una determinada familia.

  También es interesante combinar los tipos de características, en base a lo que se quiera predecir.

Kadam y col.[19] entregan una amplia información acerca de estas y características comúnmente usadas en *machine learning*

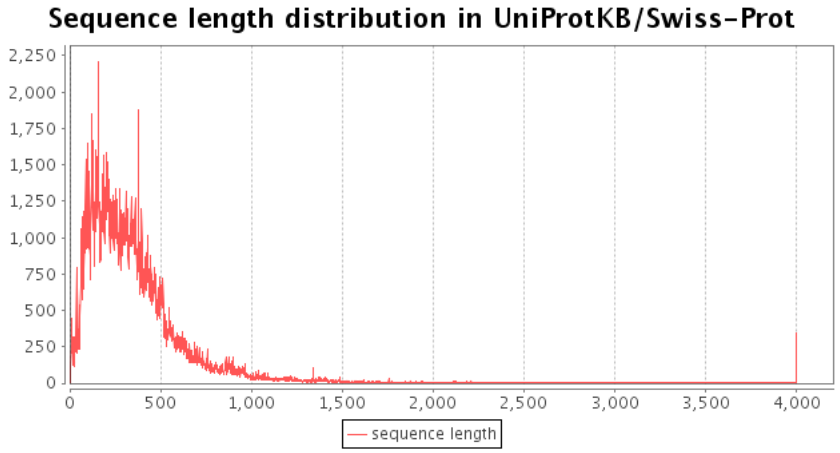


Figura 2.8: Distribución del largo de las secuencias en UnitProtKB/Swiss-Prot. Imagen extraída de [5].

[5] http://www.uniprot.org/statistics/

2.3.2. Escalar datos

El escalar los datos se hace necesario pues las distintas características extraídas no siempre se encuentran en las mismas dimensiones, por ejemplo, una en metros y otra en centímetros. Dado que la mayoría de los algoritmos se basan en nociones de distancias, esta debe ser medida siempre dentro de un mismo rango, de lo contrario habrá un sesgo hacia una de ellas.

Una de las maneras más comunes de escalar es estandarizar los datos. Esto significa convertir los datos para que tengan media cero y varianza uno. Una alta varianza podría dominar la función objetivo a minimizar.

2.3.3. Selección de características

Ante la presencia de datos de alta dimensionalidad, el costo computacional aumenta y los modelos pueden sobre-ajustarse. El sobreajuste tiene relación con el aprendizaje específico de los datos de entrenamiento que no son representativos de la estructura principal de los datos. De esta manera, se aprenden relaciones únicas del conjunto de entrenamiento y se es incapaz de generalizar el aprendizaje a nuevos datos [20]. Ello debido a la alta cantidad de información manipulada, que no siempre aporta, puesto que puede corresponder a información redundante e irrelevante ante el problema en sí. Así, características irrelevantes que no están asociadas directamente a las clases buscadas afectan el proceso de aprendizaje. Es por ello que se hace necesario aplicar métodos de procesamiento de datos para obtener un subconjunto de características que sean realmente informativas, permitiendo al modelo aprender y tener capacidad generalizadora, aumento su rendimiento y disminuyendo el costo de cálculo computacional. El conjunto de técnicas que se encarga de eso son llamadas selectores de características, los cuales pertenecen a un grupo aún mayor de técnicas que se encargan de reducir dimensionalidad, ya sea seleccionando o combinando los datos en bruto. Típicamente, estos métodos se agrupan en tres clases: filtrado, *wrapper* y *embedded* [21]. Solo se detallará el primero.

Métodos por filtrado

La selección de características por medio del uso de métodos de filtrado se basan en la selección de características de manera independiente al clasificador, cluster o regresor utilizado, solo relacionando estas con la variable a predecir (etiquetas), mediante medidas como distancia, dependencia, información o correlación. Tienen la ventaja de ser mucho menos costosos que los métodos *wrapper* los cuales no se pueden aplicar en alta dimensionalidad (tiempos de cálculo excesivos).
Usualmente consta de dos etapas: primero, se genera un ranking de características basado en algún criterio. Cada característica puede ser rankeada individualmente (*univariate*) o considerando la interacción con las demás (*multivariate*). Luego, se escogen las de mejor ranking (la cantidad es un parámetro a ajustar) para ser usada en el modelo clasificador.
Dos de los acercamientos más comunes para generar ranking individualmente (*univariate*) corresponden a :

- F-Test: El test de Fisher se basa en la idea de que características de alta calidad asignan valores

similares a muestras de la misma clase, y valores distintos para muestras de otras clases. El puntaje de la característica i, S_{i}, corresponde a :

$$S_{\mathrm{i}} = \frac{\sum_k n_j (u_{\mathrm{i}j} - u_{\mathrm{i}})^2}{\sum_k n_j \rho_{\mathrm{i}j}^2} \tag{2.2}$$

Con $u_{\mathrm{i}j}$ y $\rho_{\mathrm{i}j}$ la media y varianza de la característica i en la clase j respectivamente, n_j el número de muestras en la clase j y u_{i} la media de la característica i.

- Información Mutua (MI) : Mide la dependencia entre variables. Cero indica que son variables aleatorios independientes, mientras que a mayor valor hay una mayor dependencia. Se basa en la estimación de la entropía en cierta vecindad de datos. Sean dos conjuntos U y V con particiones U_{i} y V_j. $P(\mathrm{i})$ la probabilidad de que una muestra este en el conjunto U_{i} y $P'(j)$ la probabilidad de que una muestra esté en el conjunto V_j. La información mutua entre U y V esta dado por :

$$MI(U,V) = \sum_{\mathrm{i}} \sum_{j} P(\mathrm{i},j) \times \log \frac{P(\mathrm{i},j)}{P(\mathrm{i})P'(j)} \tag{2.3}$$

  Esto es igual a la divergencia de Kulback-Leibler [22] de la distribución conjunta con el producto de las distribuciones marginales.
  Al usar este método, el conjunto U se ve representado por la característica i-ésima de todo el conjunto de entrenamiento y las etiquetas por el conjunto V. Así, mediría la dependencia entre cada una con las clases a asignar y generaría un ranking.

La diferencia de ambos métodos radica en que el primero busca una relación lineal entre las características y las etiquetas, mientras que el segundo busca cualquier tipo de relación (no lineal) entre ellas. Esto hace del segundo método más complejo computacionalmente.

Una vez generado el ranking se debe escoger el número de características a utilizar. Esto se puede hacer con los siguientes métodos:

- Seleccionando un porcentaje p de las características.
- Seleccionando las k mejores características.

2.3.4. Clasificadores

El problema de clasificar consiste en identificar a que categoría pertenece un nuevo dato obtenido, basado en lo observado en el período de entrenamiento. Al conjunto de algoritmos que realizan esta tarea se les llama clasificadores. A continuación se detallan algunos de los más comunes

K-Nearest Neighbors

K-Nearest Neighbors [23] es un tipo de clasificador de aprendizaje no generalizado; es decir, no intenta construir un modelo con los datos, sino que almacenarlos para poder compararlos a futuro. La clasificación se lleva a cabo mediante una votación de los K vecinos más cercanos, en donde la clase más representada entre ellos es la asignada. Para un valor grande de K se obtiene una robustez ante valores

atípicos, lo que puede significar un problema si esos valores deben ser considerados. Para obtener los K vecinos más cercanos a un punto se necesita una métrica de distancia, siendo la euclidiana la más común. Normalmente los K vecinos tienen el mismo peso a la hora de votar. Sin embargo, pueden existir ocasiones en donde se necesite dar más peso a los que estén más cerca.

Support Vector Machines

Support Vector Machines [24] es un tipo de clasificador supervisado, el cual construye un hiper-plano que permita separar las distintas clases. Se busca que este hiper plano tenga la máxima distancia posible a las muestras de entrenamientos más cercanos (maximizar el margen). La fig. 2.9a ejemplifica un caso en que los datos de entrada son separables por un plano. Cuando los datos de entrada no se pueden separar mediante un plano, es necesario llevarlos a otra dimensión en donde sí se pueda (fig. 2.9b). Esto se hace mediante una función llamada kernel, la cuál mide similitud entre dos muestras. Sin embargo la medida de similitud se puede computar en otro espacio de alta dimensión si se complejiza la expresión de distancia. Existen distintos tipos de kernels: lineal, polinomial o RBF (*radial basis function*) gaussiana, entre los más comunes.

- lineal = $< x_{\mathrm{i}}, x_j >$
- polinomial = $(\gamma < x_{\mathrm{i}}, x_j > +r)^{\mathrm{d}}$ con d el grado y $r \geq 0$ un parámetro libre.
- Radial basis function = $\mathrm{e}^{-\gamma|x_{\mathrm{i}}-x_j|^2}$ con $\gamma = \frac{1}{2\sigma^2}$ que mide la influencia que tiene cada muestra.

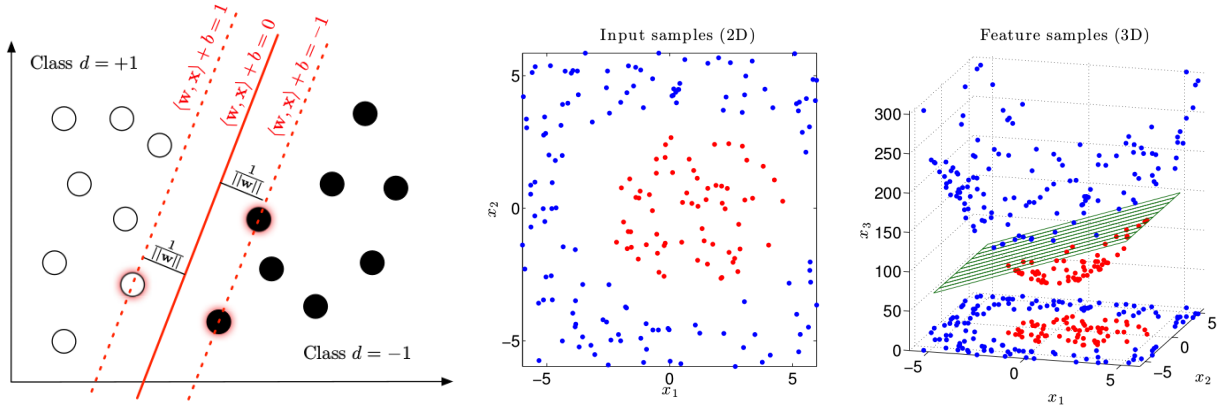
Si se tiene un conjunto de entrenamiento x_{i}, con $\mathrm{i} = 1..N$ distribuido en dos clases (clasificación binaria) $y \in \{1, -1\}^n$, el problema dual a resolver es :

$$\begin{aligned} \underset{\alpha}{\text{minimizar}} \quad & \frac{\alpha^T Q \alpha}{2} - \mathrm{e}^T \alpha \\ \text{sujeto a} \quad & y^T \alpha = 0 \\ & C \geq \alpha_{\mathrm{i}} \geq 0 \qquad \mathrm{i} = 1.. \end{aligned} \tag{2.4}$$

Con e un vector de 1's, $C > 0$ una cota superior que representa el costo de clasificar incorrectamente, Q es una matriz semidefinida positiva tal que $Q_{\mathrm{i}j} = y_{\mathrm{i}} y_j K(x_{\mathrm{i}}, x_j)$ con $K(x_{\mathrm{i}}, x_j) = \phi(x_{\mathrm{i}}^T)\phi(x_j)$ el kernel. ϕ es la función que lleva a los datos a una alta dimensión. Aquellos x_{i} para los cuales su α_{i} asociado es distinto de cero se le llama vectores de soportes. Son aquellos que se ubican sobre el margen de la región de decisión como se indica en la fig. 2.9a. Las muestras de entrenamiento que no sean vectores de soporte se vuelven irrelevantes. La función de decisión ante una nueva muestra corresponde a :

$$sgn(\sum_{\mathrm{i}=1}^{n} y_{\mathrm{i}} \alpha_{\mathrm{i}} K(x_{\mathrm{i}}, x) + \rho \tag{2.5}$$

ρ se calcula internamente a partir de la condición que cumplen los vectores de soporte. La función sgn dará el signo de la expresión, lo que dará la clase correspondiente.



(a) Separación de clases por medio de un plano

(b) Cuando no es linealmente separable en el espacio de entrada

Figura 2.9: Support Vector Machine

Para manejar casos multiclases, existen dos esquemas posibles : *one vs one* o *one vs rest*. En el primer caso, si existen k clases, estas se deben comparar con las $k - 1$ restantes. Para ello se crean $\frac{k \times (k-1)}{2}$ clasificadores binarios, en donde la decisión final se toma mediante votación de todos ellos. En el segundo caso, se crean k clasificadores, en donde en cada uno se enfrenta a cada clase con todas las restantes, considerando a estas últimas como pertenecientes al mismo grupo. Ante una muestra a predecir, la clase asignada será aquella que reporte mayor puntaje internamente.

Random Forest

Random Forest [25] es un algoritmo que ensambla un número determinado (escogido previamente) de árboles de decisión. Cada árbol por si solo predice la clase a la que pertenece la nueva muestra, para luego realizar una votación entre todos (hay variantes que promedian la predicción probabilística de cada árbol). La ventaja es que al tener una gran cantidad de árboles se disminuye el riesgo de sobreajuste que sufren los árboles de manera independiente.

Cada árbol de decisión corresponde a un modelo en donde los nodos internos contienen alguna de las características que representan a las muestras y las hojas corresponden a las posibles clases. En cada iteración, se escoge un subconjunto de los datos junto a un subconjunto de características (ambas elecciones con repetición) y se intenta ajustar un árbol. Un árbol que intente ajustar todo podría crecer en exceso con riesgo de sobreajustarse. El número máximo de características a escoger es un parámetro a elegir: con un alto número entrega diversas opciones de característica a un árbol pero a la vez disminuye la diversidad de ellos. Con una cantidad pequeña, cada árbol estaría usando una pequeña parte de las características por lo que podría encontrar relevancias locales de estas.

El número de árboles a utilizar y la cantidad mínima de muestras para separar un nodo son otros dos parámetros que se necesitan ajustar. En particular, el último controla la robustez frente a *outliers* que podría haber en los datos. El criterio para escoger que característica usar para separar un nodo se basa en aquella que sea más informativa. Esto se puede medir mediante el uso de entropía o una métrica especial

llamada *Gini impurity*. Sin embargo la una respecto la otra no presentan ventajas considerables.

2.3.5. Selección de modelos

Métricas para clasificación

Para la elección de un modelo o la validación de este, se requiere escoger una métrica que indique su rendimiento. Algunas de las métricas más comunes corresponden a :

- *Exactitud*: Calcula la fracción de predicciones correctas. Va entre 0 y 1, con 1 el mejor puntaje. Esto es:

$$acurracy(y_{\mathrm{i}}, \hat{y}_{\mathrm{i}}) = \frac{1}{n_{samples}} \sum_{\mathrm{i}=0}^{n_{samples}-1} I(y_{\mathrm{i}} = y_{\mathrm{i}}) \tag{2.6}$$

  Con y_{i} la etiqueta correcta, $\hat{y}_{\mathrm{i}}$ la etiqueta predicha, $n_{samples}$ el total de muestras y I la función indicatriz. Esta métrica es la más usada, pero en casos de clases desbalanceadas presenta el problema de que si un modelo se está midiendo con esta métrica y la quiere optimizar, entonces le colocará a cada muestra la etiqueta de la clase de mayor número de muestras. De esta manera, siempre tendrá predicciones correctas en la clase de mayor volumen y siempre se equivocará en las de menor volumen; pero dado el desbalance existente, la cantidad de predicciones correctas hará que la métrica tenga una valor cercano a 1.
- *Precisión*: Es la habilidad de disminuir los falsos positivos de una clase. Un falso positivo ocurre al etiquetar una muestra como perteneciente a una clase si no lo es. Se calcula como:

$$precision = \frac{tp}{tp + fp} \tag{2.7}$$

  Donde tp son los verdaderos positivos y fp los falsos positivos.
- *Sensibilidad* : Es la habilidad de disminuir los falsos negativos de una clase. Un falso negativo ocurre al no etiquetar una muestra como perteneciente a una clase cuando si lo es. Intuitivamente es la capacidad de encontrar todas las muestras que pertenecen a una clase. Se calcula como:

$$recall = \frac{tp}{tp + fn} \tag{2.8}$$

  Donde fn son los falsos negativos.
- *Puntaje F1*: Corresponde a la media armónica entre precision y recall, considerando ambos valores al momento de entregar un puntaje. Se puede ver como la media ponderada entre ambas métricas, pues se puede escoger ponderar una más que la otra. Se calcula como :

$$F_\beta = \frac{(1 + \beta^2) precision \times recall}{(\beta^2 \times precision) + recall} \tag{2.9}$$

  Siendo β el parámetro que controla el peso que se le da a cada término. Si se escoge igual a 1, se considera precision y recall por igual. Esta métrica, al considerar tanto fp como fn se vuelve útil a la hora de enfrentar un desbalance de clases, puesto que impide que todo se clasifique como la clase de mayores muestras, puesto que eso disminuiría su precision y por ende, su puntaje F1.

Para el caso multiclase, si se quiere entregar un puntaje F1 que resuma el comportamiento del modelo, se debe escoger alguna manera de agrupar las métricas F1 individuales por clase. Para eso se puede escoger:

- *macro* = Calcula la métrica F1 para cada clase. Luego, promedia todas sin considerar el desbalance entre clases que pueda existir.
- *weighted* = Calcula la métrica F1 para cada clase, y luego la promediarla, pondera cada una por la cantidad de muestras en la respectiva clase. De esta manera, considera más difícil clasificar bien toda una clase con muchas muestras más que una pequeña, por lo que le otorga un mayor peso al puntaje obtenido en aquel caso.

Validación cruzada

Entrenar y probar con los mismos datos es un error pues el modelo tan solo tiene que repetir las etiquetas que le fueron entregadas en el entrenamiento y con eso obtendría un 100 % de rendimiento. Es por ello que una práctica común es separar el conjunto de datos en uno para entrenar y otro para probar (ver la capacidad generalizadora del modelo en datos que no ha visto). Aún así se corre el riesgo de que el modelo se sobreajuste si se elige el conjunto de parámetros que mejor lo hace con los datos de prueba, puesto que es un conjunto escogido de manera aleatorio, por lo que puede pasar que solo se encuentren las muestras fáciles de clasificar y los parámetros encontrados se ajustan a ellos. Es por ello que se agrega un tercer conjunto: el de validación, el cual cumple la función de control de calidad frente al ajuste del modelo en entrenamiento. Para cada modelo ajustado, se prueba en el conjunto de validación. Se escoge el que dé mejor resultado para luego probar su capacidad generalizadora en el conjunto de prueba. De esta manera, si los parámetros se llegaran a sobreajustar para tener un buen rendimiento solo en el conjunto de validación, su capacidad generalizadora sería baja y se vería en el conjunto de prueba.

Desafortunadamente, esto requiere particionar el conjunto de datos en tres partes, lo cual disminuye considerablemente la cantidad de datos para entrenar, en especial en casos de pocas muestras. Es ahí cuando la validación cruzada nace como alternativa. Esta consiste en dividir el conjunto de datos en k partes, para entrenar con las k-1 y probar con la restante. Cada una de las partes le corresponde usarse como conjunto de prueba una vez. De esta manera se elimina el conjunto de validación. Además, como cada conjunto es usado una vez como prueba, permite que el modelo que mejor se ajuste sea aquel que tiene el mejor rendimiento en todos los datos. El rendimiento se obtiene como el promedio de los puntajes de cada iteración. Existe diversas maneras de particionar las k partes ([26]). Algunas son :

- K-Fold : Se realiza al azar
- Stratified K-Fold: Busca mantener la proporción de clases en cada una de las partes. Especial para casos de desbalance de clases, puesto que de lo contrario podría quedar toda la clase de menos representantes en una de las partes, por lo que cuando le toque usarse como conjunto de prueba, no se habrá entrenado con ninguna muestra de esa clase y el resultado será de un 0 % de aciertos.
- Leave One Out : Caso particular cuando se escoge k igual a la cantidad de datos. Se deja una muestra afuera a la vez para probar. Tiene la ventaja de que entrega un resultado más acertado de cuál será el comportamiento del modelo a futuro, pero requiere un cálculo computacional mucho mayor. Es por ello que se utiliza cuando se tienen pocos datos, prefiriéndose alguna de las alternativas anteriores cuando se manejan volúmenes de información.

Cuando no solo se quiere estimar el error de generalización de un modelo, sino también buscar los parámetros óptimos, una técnica comúnmente usada es la validación cruzada anidada (*Nested Cross Validation*). Esta consta de dos ciclos: uno interno y otro externo, los cuales pueden ser cualquiera de las estrategias mencionadas anteriormente para particionar los datos, de manera independiente.
En el ciclo externo, se dividen los datos en k partes, teniendo k-1 para entrenar y uno de prueba. Dentro de los datos de entrenamiento ocurre el ciclo interno, en donde se vuelven a dividir, generando un conjunto de entrenamiento y uno de validación. Con el de entrenamiento se ajusta el modelo para cada combinación de parámetros buscados. Luego, se prueba su rendimiento con el conjunto de validación. Una vez se obtiene la combinación de parámetros que mejor se ajusta al conjunto de validación, se usa el conjunto de prueba para estimar el error de generalización que tiene el modelo. Esto permite no sobre ajustar el modelo a los datos de validación en el ciclo interno: si solo se usara una validación cruzada tradicional, se estaría entregando un puntaje optimista puesto que se escogió el mejor modelo para una partición específica de los datos. Para tener una idea más realista de cómo se comportará, hay que someterlo a diferentes particiones [27] . La fig. 2.10 ejemplifica lo anteriormente dicho.

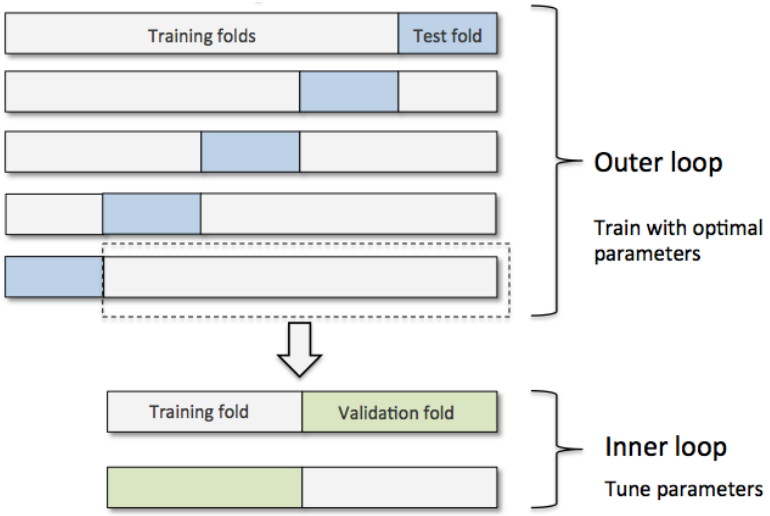


Figura 2.10: Modo de operación de la validación cruzada anidada. Imagen extraída de [2].

2.4. Predicción de funcionalidad de proteínas basados en Machine Learning

2.4.1. En la literatura

Existe una gran variedad de técnicas utilizadas para abordar el problema de predecir función de proteínas. Algunos se han basado en el uso de información de diversas bases de datos: EnzML [11] utiliza solamente descriptores de InterPro y KNN con $K = 1$ para obtener un valor de 98 % para la norma F1 micro sobre un conjunto de 300.000 enzimas. Su fuerza radica en la representación compacta usando descriptores de InterPro y en una búsqueda local que se basa en el uso de un conjunto de datos amplios para tener muchas opciones de búsqueda. DomSign [28] intenta mejorarlo utilizando un *pipeline* que incluye diccionarios y descriptores de solo una base de datos (Pfam); sin embargo, no logra mejorar los

resultados de EnzML. EFICAz también lo intenta, combinando información de distintas fuentes como PROSITE, Pfam, comparación de secuencias y uso de residuos funcionalmente discriminatorios. [15] utiliza Random Forest para predecir hasta el último número EC de enzimas que tengan una estructura 3D disponible en CATH, lo que se sabe entrega información más acertada de la funcionalidad.

Por otro lado, hay una variedad de trabajos que se han basado en el uso de características extraídas de la secuencia aminoácidida, sin embargo, no todos ellos apuntan a clasificar enzimas, sino más bien una clasificación funcional de proteínas de acuerdo a distintos criterios. De ellos, SVMProt [14] utiliza un clasificador SVM que recibe como entrada características fisicoquímicas codificadas dentro de descriptores CTD. Este método logra predecir hasta el 2do número EC en algunos casos. SVMHL [29] predice números EC hasta el tercer dígito usando descriptores CTF (*conjoint triad features*) que describen la frecuencia de aparición de triadas de aminoácidos mezcladas al uso de un alfabeto reducido que permite tratar como iguales aminoácidos distintos, pero con potencial de sustituirse mutuamente. Utilizan una variante de SVM llamada SVM *structured output* que permite realizar una clasificación jerárquica. Limita en 50 el número de integrantes por clase.

Técnicas de Deep Learning [30] también han sido probadas, sin embargo, ninguna de ellas ha intentado clasificar utilizando los números EC, sino en familias de proteínas. Cuando se habla de familias, se hace referencia a una clasificación global entre proteínas que comparten un origen común, teniendo 4 niveles de especificidad. [31] es el trabajo más avanzado, que clasifica en 689 clases de los 3 primeros niveles de especificidad a un conjunto de 5000 proteínas de prueba usando un red neuronal convolucional y una codificación one-hot [6]. Estos acercamientos requieren gran cantidad de datos, por lo que aún es un desafío llevarlo a casos como los números EC dada la distribución que tienen estos. De hecho, el trabajo mencionado no utiliza clases con menos de 150 muestras.

2.4.2. Critical assessment of functional annotation

Critical assessment of functional annotation [3] (CAFA) es un desafío impulsado por primera vez durante los años 2010-2011 para evaluar los distintos métodos de anotación automática de funcionalidad de proteínas bajo un mismo *benchmark* y poder sacar conclusiones del estado del arte actual. Durante 2013-2014 se realizó el segundo y más actual desafío CAFA (el tercero está en proceso). El desafío consiste en que se entrega un set de proteínas sin funcionalidad asociada y los diversos equipos participantes predicen su función usando sus métodos desarrollados. Cada equipo puede entrenar y usar los datos que estime convenientes. Si bien este problema utiliza otra manera de clasificación (*gene ontology*) pues se relaciona a proteínas, no solo enzimas, es de utilidad entender las herramientas que se están usando hoy en día.

Una vez ocurrido el tiempo límite para enviar las predicciones, se espera cierta cantidad de meses (8 meses se esperó en CAFA 2) para que experimentalmente se encuentre la funcionalidad de la mayor parte

[6]Codificación que transforma características no-numéricas en una representación vectorial con un 1 en la posición que corresponde a dicha característica y 0 en las demás columnas. Mayor detalle en http://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-categorical-features

del set de proteínas entregado a los participantes. Una vez evaluados todos los participantes en base a dos pruebas que se realizan (medidos con la norma F1), se entregan los resultados.

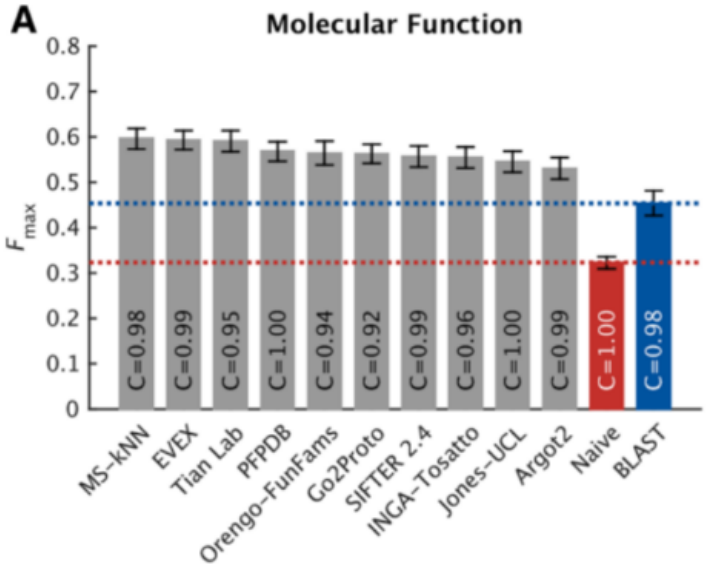


Figura 2.11: Rendimiento de mejores métodos en CAFA2 en predicción de función molecular en términos de métrica F1. C indica el porcentaje de cobertura que obtuvieron. Imagen extraída de [3]

El criterio de clasificación *Gene Ontology* consta de sub-clasificaciones internas: función molecular, proceso biológico y componente celular donde se ubica. De estas, es la primera la que tiene mayor relación a la clasificación usada en el presente trabajo (números EC). En la fig. 2.11 se muestra los métodos con mejor rendimiento en asociar funcionalidad a una proteína (C indica el porcentaje de cobertura que obtuvieron). BLAST y un método basado en contar frecuencias de aparición para predecir (*Naive*) son los algoritmos contra qué compararse. Se ve que los métodos actuales logran superarlos, teniendo un rendimiento (puntaje F1) alrededor del 60 %. Dentro de los mejores evaluados, MS-KNN utiliza un algoritmo de KNN con diversas fuentes de información (similitud, interacción entre proteínas y expresión génica), mientras que EVEX utiliza herramientas de data mining para buscar información en todo tipo de artículos disponibles en la web. Una de las razones que se entrega para entender la diferencia en rendimiento es la dependencia de conjunto de entrenamiento que usen, teniendo variabilidad en el sesgo o la especificidad de la anotación del conjunto de datos utilizado.

Ningún método sobrepasa a los demás considerablemente, mostrando que este es un campo abierto, donde no hay algún algoritmo o fuente de información sobre otros. La combinación de estos es lo que da fuerza al modelo predictivo. Esta es la conclusión más relevante del CAFA 2.

Capítulo 3

Herramientas utilizadas

En el presente capítulo se detallan las herramientas utilizadas para la implementación del algoritmo propuesto, así como para el pre-procesamiento de los datos.

3.1. Lenguaje de programación

Las herramientas a utilizar son librerías escritas en *Python*. *Python* es un lenguaje de programación interpretado que soporta entre otras cosas programación orientada a objetos y programación imperativa. Posee una licencia de código abierto. Dentro de librerías a utilizar se encuentran :

- Biopython [32] : Es un proyecto colaborativo internacional de código abierto que provee librerías para problemas de bioinformática. Permite leer y manejar secuencias en distintos formatos, además de comunicarse directamente con herramientas como BLAST.

- SciPy [33]: Colección de herramientas para diversos usos científicos. Incluye **NumPy** para computación numérica, **Matplotlib** para realizar gráficos, **Pandas** para un manejo de alto eficiencia de estructuras de datos, entre otros.

- Scikit-Learn [34] : Una de las librerías de *machine learning* más completas que existen. Comprende simples y eficientes herramientas para minería y análisis de datos. Es de código abierto y está construida sobre otras librerías comúnmente usadas como **NumPy**, **Matplotlib** y **SciPy**. Es usada por empresas como Spotify, Booking o Evernote, entre otras.

- Protein Feature Engineering Toolkit [35] (ProFET): *package* desarrollado en *Python* para la extracción de numerosas características de proteínas, lo que incluye algunas ampliamente usadas en la literatura como otras propuestas por el autor. Es de fácil uso y contiene funcionalidades desarrolladas sobre *SciPy* y *Scikit-Learn* por lo que su incorporación resulta cómoda puesto que son las herramientas escogidas para este trabajo.

  Las categorías en que se encuentran las características disponibles son las siguientes:

- Propiedades biofísicas : La mayoría ha sido validada en [36].
- Características basadas en letras : Composición aminoacídica, n-gramas y uso de alfabetos reducidos, entre otros.
- Características potencialmente locales : *Potencial post-translational modification sites* (PTM) y *Potencial disorder*
- Estadísticas basado en entropía de información : Entropía por letra, autocorrelación binaria, entre otros
- Descriptores CTD: *Composition*, *Transition* y *Distribution*. Desarrolladas por [37] y usados en diversos trabajos como [10]

El detalle completo de todas las características junto a una descripción de cada una se puede encontrar en [35].

3.2. CD-HIT

CD-HIT [38] es un programa para agrupar secuencias biológicas y con ello reducir redundancia. Está diseñado para manejar grandes conjuntos de datos y generar resultados de manera rápida debido a su paralelización. Incluye diferentes modos de uso como CD-HIT, CD-HIT-2D, CD-HIT-454, entre otros, siendo el primero de estos el de interés para este estudio. CD-HIT agrupa proteínas similares en grupos de datos que sobrepasan un umbral de similitud definido por el usuario.

3.3. Diamond

Diamond [39] es un alineador de secuencias que surge como alternativa a BLAST debido a su eficiente implementación (x20.000 la velocidad de BLAST), aunque utilizando internamente los mismos algoritmos que este.

3.4. InterPro

InterPro [40] es una plataforma que provee análisis funcional de proteínas, clasificándolas en familias y prediciendo la presencia de dominios y sitios importantes. Tiene acceso a diversas bases de datos (mostradas en la tabla 3.1) que forman *The InterPro Consortium*, las cuales clasifican proteínas usando tanto información manualmente curada como herramientas bioinformáticas. Cada base de datos genera clasificaciones en base a sus criterios; para cada proteína, esta será clasificada de acuerdo a la nomenclatura que utilice cada base de datos. Por ejemplo, una proteína puede estar clasificada como PF00106 en Pfam, PS01159 en PROSITE y PTHRE43157:SF1 en PANTHER, por mostrar algunas bases. InterPro recoge toda esa información y la entrega a modo de descriptores por cada proteína. Adicional a esto, la plataforma ofrece la posibilidad de generar sus propios descriptores, unificando la información obtenida de las distintas fuentes. Si en el ejemplo anterior, la plataforma considera que la clasificación PF00106 y PS01159 hacen referencia al mismo grupo de proteínas, entonces mezcla ambas y les asigna un nuevo

Base de datos	Descripción	Base de datos	Descripción
CATH-Gene3D	Clasificación en familias basado en la localización de dominios estructurales	CDD	Anotación de proteínas basado en múltiples alineamientos de secuencias para antiguos dominios
MobiDB	Anotaciones de proteínas intrínsecamente desordenadas. Contiene información manual y predicha	HAMAP	Anotaciones manuales de proteínas basado en familias bien conservadas
PANTHER	Colección de familias de proteínas divididas en subfamilias relacionadas, usando conocimiento experto	Pfam	Colección de múltiples alineamientos de secuencias
PIRSF	Clasificación de proteínas que refleja relaciones evolucionarias de estas usando su largo total y dominios	PRINTS	Enciclopedia de grupos de motivos conservados que permiten caracterizar familias o dominios
ProDom	Base de datos de dominios	PROSITE	Clasificación en familias en base a sitios biológicamente importantes, patrones y perfiles
SFLD	Clasificación jerárquica de enzimas que relacionan características estructurales con capacidades químicas	SMART	Provee análisis de dominios
SUPERFAMILY	Clasificación estructural basado en dominios	TIGRFAMs	Herramienta para identificar proteínas relacionadas funcionalmente basado en homología

Tabla 3.1: Descripción de bases de datos de *The InterPro Consortium*

descriptor IPR0001. Estos descriptores IPR son propios de la plataforma, los cuales tienen la ventaja de combinar información, entregando una manera más compacta y no redundante de describir una proteína. Cada una de estas puede tener uno o más descriptores IPR, así como no todas las bases de datos son capaces de agruparse bajo una sola etiqueta IPR, debido a que hacen referencia a clasificaciones distintas.

La utilidad de InterPro radica en la diversidad de fuentes que utiliza, produciendo una poderosa y compacta representación de las proteínas. Además, incluye información manualmente revisada. En su última versión (v64), InterPro mantiene una alta cobertura sobre todas las proteínas existentes en UnitProKB. La figura 3.2 muestra los porcentajes de descriptores existentes para TrEMBL y SwissProt. Para el caso SwissProt, solo un 3 % de los datos no tienen ningún tipo de descriptor asociado, lo que es poco y en constante decaimiento. Se ve que los descriptores IPR tienen una menor cobertura, dado el trabajo que conlleva analizar los descriptores de todas las demás fuentes e intentar combinarlas en uno solo.

InterPro se puede utilizar tanto de manera local como remota. Cuenta con una herramienta llamada InterProScan [41] la cual permite escanear una secuencia y generar sus descriptores en las bases de datos que uno escoja. Esta se actualiza cada 2 meses, incorporando información que ha sido recogida en las distintas fuentes en que se basa.

Base de datos	Versión	Cantidad de proteínas	Descriptores	
			Cualquier base	*IPR*
TrEMBL	2017_07	88032926	77578215 (88.1 %)	70844347 (80.5 %)
SwissProt	2017-07	555100	543462 (97.9 %)	536222 (96.6 %)

Tabla 3.2: Porcentaje de cobertura de InterPro en bases de dato de UnitProtKB

Capítulo 4

Metodología e Implementación

En este capítulo se detallan las diversas pruebas que se hicieron previamente a modo de búsqueda de alternativas así como acercamientos al problema en sí, para luego dar paso a la propuesta a desarrollar.

4.1. Primeras pruebas

Inicialmente, se adquirió un conjunto de datos de SwissProt (versión 2017_04) que representan 20 distintas clases EC representadas hasta el tercer dígito, las cuáles se ha visto experimentalmente que son difíciles de discernir entre ellas mediante similitud de secuencias. Consisten en un total de 14.991 muestras. El detalle por clase se despliega en la tabla 4.1.

Clase EC	**Número de muestras**	**Clase EC**	**Número de muestras**
3.2.1	2130	1.10.3	407
2.1.1	2002	2.4.2	355
2.4.1	1892	1.14.14	353
2.3.1	1857	1.13.11	301
1.1.1	1726	5.4.99	160
3.1.1	1596	4.1.2	118
1.11.1	523	4.3.1	104
1.14.13	435	1.14.99	89
1.14.11	434	5.3.99	57
1.2.1	423	1.14.21	32

Tabla 4.1: Distribución de las 20 clases a analizar. Extraídas de UnitProtKB/Swiss-Prot 2017_04 release y luego escogidas manualmente por un experto.

En un principio, se busca evaluar distintos métodos de representar la secuencia aminoacídica como un vector de largo fijo. Para ello, se compara el uso de ProFET e InterPro en base al rendimiento obtenido por clasificadores sobre el conjunto de datos previamente mencionados; es decir, a la métrica F1 *macro*

obtenida. Se realiza hasta el 3er EC ya que las clases tienen un número considerable de muestras lo que permitiría evaluar el comportamiento de ambos acercamientos sin la exigencia que demanda llegar al cuarto dígito EC. No se usaron los descriptores propios de InterPro (IPR) en esta prueba.

4.1.1. Uso de ProFET

Como primer paso, cada enzima se representa por medio de un vector de 1170 características que incluye todas las posibles opciones disponibles en ProFET, por lo que se obtiene una matriz de (14991,1170); 14991 proteínas, cada una representada por un vector de características de tamaño 1170. El algoritmo que se utiliza consta de las siguientes etapas:

- Separar datos : Para poder entrenar y validar el modelo se deben separar los datos. Con el conjunto de entrenamiento se ajustan los parámetros necesarios para luego probar dicho modelo en datos no vistos. Es muy importante hacer esta separación pues si se mezclaran ambos conjuntos se podría llegar a predecir información que ya se analizo durante la fase de entrenamiento llegando a resultados sobrestimados. Este problema es conocido como *Data leakage* [1]. 70-30 % es una proporción comúnmente usada para separar datos de entrenamiento y validación, respectivamente. Se asegura que la proporción en que se encuentran las clases se mantenga en ambos conjuntos para evitar un desbalance excesivo hacia las clases más representadas.
- Escalar características: Se usa StandardScaler para asegurar que cada característica está centrada en 0 y con varianza 1.
- Selección de características : Por simplicidad se realiza un F-test y se escoge un determinado porcentaje de las características mejor puntuadas. Este porcentaje es parte de los parámetros que se ajustan en el entrenamiento.
- Clasificador: SVM RBF y Random Forest se prueban por ser los más utilizados en la literatura revisada.
- Métrica: Dado el problema de desbalance de clases presente, se escoge la norma F1 en su versión *macro* para darle igual importancia a todas las clases. Esta norma es una de las utilizadas en CAFA para evaluar.

Este se muestra en la fig 4.1

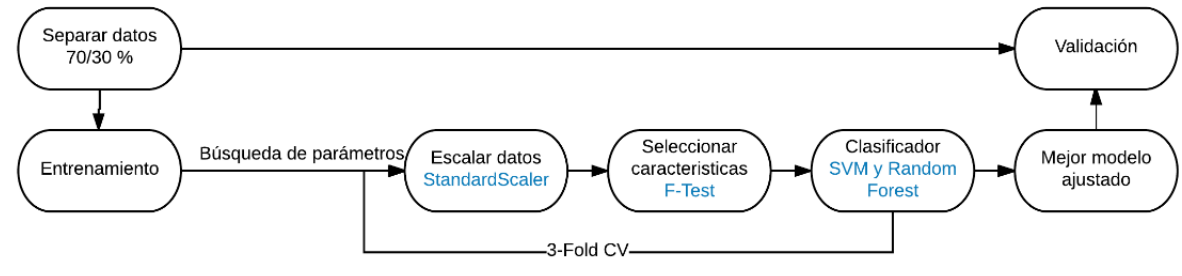


Figura 4.1: Diagrama de flujo de *pipeline* implementado

[1] http://machinelearningmastery.com/data-leakage-machine-learning/

Se realiza una búsqueda exhaustiva de la mejor combinación de parámetros para cada caso mediante una validación cruzada sobre el conjunto de entrenamiento con la clase **GridSearchCV** de *scikit-learn*. Se realizan 5 corridas para cada clasificador. A partir de eso se obtuvieron los resultados presentados en la tabla 4.3. La mejor combinación de parámetros encontrados para cada clasificador se muestra en la tabla 4.2, los cuales se obtienen como aquellos que más se repitieron en las 5 corridas.

Clasificador	**Percentile**	**Parámetros**
SVM	70 %	C = 10 γ = 0.001
Random Forest	80 %	max_features = 40 min_samples_split = 2 n_estimadors = 500

Tabla 4.2: Parámetros óptimos encontrados para cada estimador utilizando ProFET como extractor de características.

Clasificador	**F1 macro**
SVM	90.05 / 2.75 %
Random Forest	78.45 / 4.99 %

Tabla 4.3: Promedio y desviación de puntaje F1 macro obtenido por cada clasificador al usar ProFET como extractor de características.

4.1.2. Uso de InterPro

Al extraer descriptores de InterPro, se pasa de usar características continuas a discretas. Para generar el vector de largo fijo que describirá cada enzima, se obtiene el total de descriptores distintos existente en las 20 clases. Estos son 7368, lo que da el largo del vector que describe cada muestra. Cada descriptor corresponde a una columna del vector, la cual tendrá un 1 si dicha enzima lo contiene o 0 de lo contrario. Con esto se genera una matriz de (14991, 7368). El algoritmo utilizado es el mismo que se mostró en la fig 4.1, con la diferencia de que ya no es necesario estandarizar las características (todas están en el mismo rango) por lo que se remueve esa etapa. Además, se incluye KNN como clasificador. En la tabla 4.5 se ve el aumento de rendimiento en todos los clasificadores, lo que muestra la ventaja de los descriptores de InterPro frente a las características de ProFET. KNN y SVM obtienen los mejores puntajes, sin diferencias notorias entre ellos. Por otro lado, en tabla 4.4 se ve como Random Forest logra disminuir el número de árboles necesarios producto de una separación más clara entre clases dado por las nuevas características usadas, lo que se potencia con la capacidad del mismo algoritmo de generar un ranking de ellas. SVM se muestra más forzado al actuar, usando todas las características y aumentando el factor de penalización C.

Esta prueba permite concluir que el uso de los descriptores de InterPro es superior a las características de ProFET. Su ventaja radica en que el primero incorpora información de dominios funcionales y en algunos casos de estructura tridimensional, lo que se sabe que tiene una mayor relación con la funcionalidad. Las características de ProFET solo se basan en la secuencia completa, lo que puede estar siendo contraproducente pues no toda la secuencia entrega información importante, a diferencia de si se extraen solo los dominios o sitios relevantes.

Tanto KNN como SVM tienen un rendimiento similar. Ante un volumen considerable de datos, SVM es capaz de generar hiper planos bien definidos; una vez generado esto, la predicción es bastante rápida pues solo basta calcular la posición con respecto al hiperplano generado. Por otro lado, KNN es más

Clasificador	Percentile	Parámetros
KNN	75 %	k = 3 weight = distance
Random Forest	25 %	max_features = 40 min_samples_split = 2 n_estimators = 100
SVM	100 %	C = 500 γ = 0.001

Tabla 4.4: Parámetros óptimos encontrados para cada estimador utilizando InterPro como extractor de características.

Clasificador	F1 macro
KNN	95.54 / 0.85 %
Random Forest	88.12 / 0.75 %
SVM	95.82 / 0.44 %

Tabla 4.5: Promedio y desviación de puntaje F1 macro obtenido por cada clasificador cuando se utiliza InterPro como extractor de características.

lento en predecir pues debe comparar con toda la información de entrenamiento y no generaliza, lo que tendría cierta desventaja ante grandes cantidades de datos. Sin embargo, en el caso contrario, si se tienen clases con pocas muestras, este último toma ventaja pues el tiempo de predicción ya no es considerable, y mientras exista una muestra similar, este será capaz de predecir correctamente pues puede realizar una búsqueda extremadamente local (k = 1), mientras que SVM no podrá generar un hiper plano bien definido.

4.2. BLAST

4.2.1. Comportamiento en enzimas

Es necesario conocer el desempeño que tiene BLAST en SwissProt, para entender sus limitaciones y ventajas. Para ello, se obtiene la versión 2017_05 de la base de datos. Usando *CD-HIT* se filtran las secuencias repetidas que comparten más de un 99.99 % de similitud entre secuencias y el mismo largo. La razón de que existan secuencias repetidas se debe a que la misma proteína (o enzima) puede estar en organismos clasificados como distintos pero cuya diferencia es mínima, por ejemplo, entre accesiones de bacterias, por lo que se le asigna un identificador distinto. Para efectos de este estudio, se agrupan para obtener solo un representante, por lo que no se considerará la taxonomía de donde provienen.

Se extraen solo las enzimas (aquellas que tienen asignado al menos un número EC) y luego se filtran para obtener aquellos casos en que se conocen los 4 dígitos EC. Esto debido a que en su origen, este es un problema multiclase : Si una enzima pertenece a la clase EC 1.1.1.1, también pertenece a la 1.1.1.-, 1.1.-.- y 1.-.-.- . Al trabajar solo con aquellos que tienen sus cuatro dígitos EC completos, se transforma el problema a un caso de una clase por enzima. Esta es una de las restricciones propuestas basados en el trabajo de Schaläpfer, 2017 [42]. Los casos de enzimas que cuentan con más de una etiqueta se consideran una clase aparte. Es decir, si una enzima pertenece a la clase A, otra a la B y una tercera a ambas, se consideran tres situaciones distintas y no se mezclan entre ellas. La tabla 4.6 muestra la base de datos que se obtiene luego del proceso de filtrado, la cual se usará de acá en adelante. Por último, se

consideran solo las clases con al menos 2 muestras; de lo contrario, no se tendrían dos secuencias que se puedan alinear en esas clases, forzando a que el programa entregue una clasificación errónea.

	Total de muestras	**Total de clases**
SwissProt Filtrada	188.179	3.610

Tabla 4.6: Datos con los que se trabajará, obtenidos luego del proceso de filtado.

Una vez que se tienen los datos filtrados, se procede a ejecutar *Diamond* utilizando el mismo conjunto como datos de referencia; es decir, se busca clasificar cada enzima del conjunto de datos buscando la secuencia más similar a ella sin considerarse a si mismo. Se realizan diversas corridas cambiando algunos de los parámetros del programa: la sensibilidad de búsqueda y el valor de corte *e-value*. Por defecto, este usa un modo *fast* que busca alinear rápidamente y filtrar más zonas de baja complejidad, mientras que como valor de corte se usa un *e-value* de 0.001. Para explorar las otras opciones del programa, se realizan corridas utilizan el modo *more sensitive* el cual busca alineaciones con mayor sensibilidad (filtrando menos zonas de posibles alineaciones) y un *e-value* de 10, pues es el valor por defecto que usa el programa original (BLAST). En la tabla 4.7 se muestra el puntaje F1 *macro* obtenido por cada configuración, además del número de clases en que la clasificación es perfecta (F1 = 100 %), en las que no y los casos en que el programa no encontró ninguna secuencia similar; es decir, se debía predecir una clase A pero no se predijo nada. Este último caso se separa puesto que el problema no es que existan dos clases que se confundan entre ellas, sino que no existe una secuencia lo suficientemente similar para poder asignar alguna clase.

Casos	**F1 macro**	**Clases con F1 = 100 %**	**Clases con F1<100 %**	**Clases en que no pudo alinear alguna muestra**
e = 0.001 fast	88.87 %	1.969	1.641	262
e = 0.001 more sensitive	89.61 %	2.076	1.534	142
e = 10 fast	88.93 %	1.983	1.627	206
e = 10 more sensitive	89.68 %	2.048	1.562	56

Tabla 4.7: Resultados de ejecutar *Diamond* en distintas configuraciones sobre la base de datos filtrada.

El puntaje obtenido es bastante similar para todos los casos. La diferencia radica en la cantidad de secuencias que no se logran alinear dada las restricciones impuestas. Ya se había mencionado el problema con las secuencias cortas (péptidos) los cuales suelen filtrarse cuando se hacen alineaciones sin tanta sensibilidad por no considerarse una secuencia en sí, sino un fragmento. Al pasar esto, esa pequeña parte se repite en muchas otras secuencias, se considera que es una zona de baja complejidad (muy común) por lo que se elimina. En la fig 4.2 se muestran los histogramas del grado de similitud que tienen secuencias

que no se clasificaron correctamente con su mejor alineación en la base de datos. Para los casos *fast* se obtiene una mayor cantidad de secuencias no alineadas, representadas por la barra centrada en 0, que en el caso *more sensitive*. Por otro lado, un *e value* mayor ante un mismo modo de alineación también logra disminuir las secuencias no alineadas. La distribución de las secuencias que sí se alinean es similar en todos los casos salvo en el caso mostrado en amarillo, en donde hay un aumento de los casos de error cuando la similitud entre las secuencias es menor al 40 %, como se menciona en [43]. En el mismo trabajo se menciona que sobre un 60 % de similitud es una medida confiable para clasificar correctamente un 90 % de los casos, lo que se reafirma pues los casos de clasificación errónea con una alta similitud son pocos.

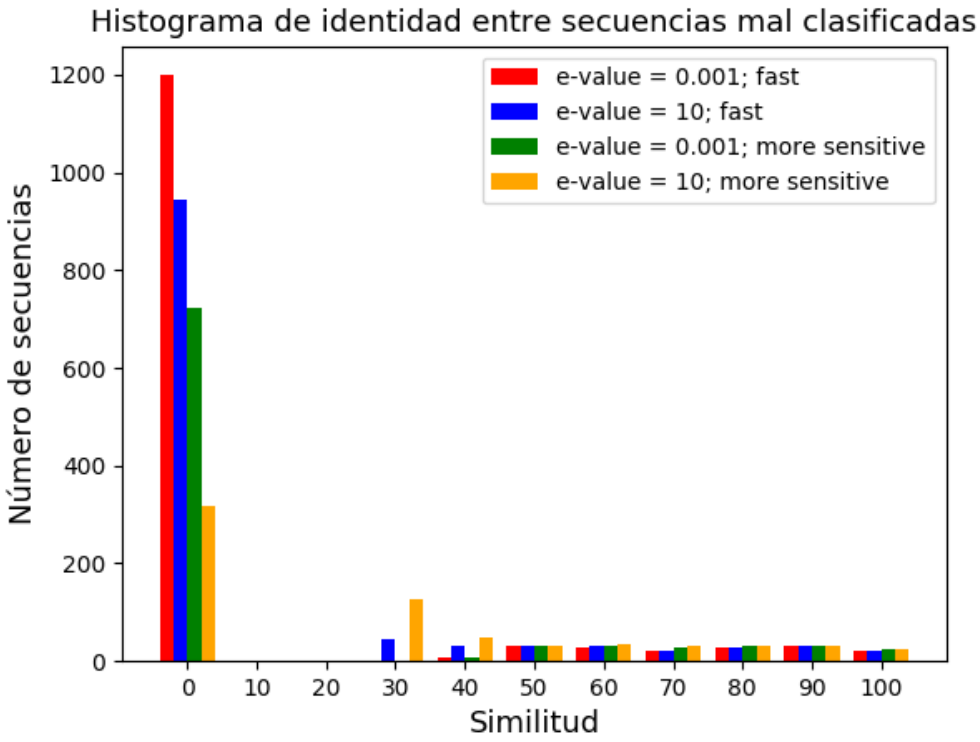


Figura 4.2: Histogramas del grado de similitud de secuencias incorrectamente clasificadas con su mejor alineación, en cada una de las distintas configuraciones experimentadas en Diamond, separadas por color. Se verifica que la configuración en amarillo obtiene una menor cantidad de secuencias no alineadas (centradas en cero) junto a una mayor cantidad de secuencias clasificadas incorrectamente con un 30 % de similitud a alguna enzima en la base de datos.

Se desprende que el modo *more sensitive* junto a un *e-value* igual a 10 entrega la mejor configuración, considerando el puntaje obtenido y la menor cantidad de casos no alineados. Se prefiere obtener una mayor cantidad de secuencias mal clasificadas, ante que no alineadas, puesto que las primeras se pueden corregir. Dada la alta cantidad de casos en que se logra una clasificación perfecta, se hace necesario incorporar la alineación de secuencias como parte del algoritmo propuesto, reduciendo el problema a los casos en donde esta falla.

Dado que los números EC representan una clasificación jerárquica, se vuelve interesante saber hasta que punto BLAST logra clasificar de manera correcta. En la tabla 4.8 se muestran la cantidad de casos en que, si bien se clasificó mal en primera instancia, si se comparan los 3 primeros números EC, se comparte la clasificación. Se ve que tanto en los casos en que las secuencias no comparten ningún número EC, como en los que comparten el primer y el segundo, BLAST tiene un rendimiento parejo, equivocándose en 100 muestras en cada caso. La gran cantidad de casos se concentra en el paso del 3er al 4to EC; es ahí en donde se encuentra una gran cantidad de casos que comparten los 3 primeros dígitos y es en el último en el que difieren. Esto indica que el rendimiento que tiene BLAST en asignar los 3 primeros EC es mucho mayor que al de asignar el último, puesto que este es el de mayor especificidad, impulsando a que el algoritmo se confunda en una mayor cantidad de veces.

En la tabla 4.9 se muestra el resultado de analizar los casos con más de una clase EC asignada. Se muestra la cantidad de veces en que BLAST entrega como mejor alineación un secuencia que tiene más o menos clases que las reales, o que si bien tienen la misma cantidad, difieren en alguna. Se desprende que el problema más común para BLAST es asignar más clases de las que realmente tiene una enzima. Esto se debe a que la secuencia mejor alineada incluye partes que no se encuentran en enzima buscada, lo que puede ser un error proveniente de la base de datos (la secuencia aminoacídica que se tiene no está completa por obtenerse de un transcriptoma) o bien puede ser que aquella enzima realmente no realice todas las funcionalidades que se le asigna.

Números EC compartidos en casos mal clasificados	**Ninguno**	**1er EC**	**2do EC**	**3er EC**	**Total**
Cantidad de casos	98	99	96	876	1169

Tabla 4.8: Cantidad de secuencias que comparten alguno de los números EC a pesar de haber sido mal clasificada. No incluye los casos con más de una clase asignada.

	BLAST asigna más clases	**BLAST asigna menos clases**	**Se asigna la misma cantidad**	**Total**
Cantidad de casos	577	206	37	820

Tabla 4.9: Comportamiento de casos con más de una clase cuando BLAST no clasifica correctamente. Se muestra cuando este asigna más, menos o la misma cantidad de clases

4.2.2. Comportamiento sobre proteínas

Un paso previo a clasificar enzimas en números EC es determinar si la secuencia de prueba corresponde a una enzima o no. Para ello, se considera la base de datos de enzimas usado previamente y se le agregan las proteínas que no corresponden a enzimas. Usando la misma configuración sobre BLAST (modo *more sensitive* y e-value igual a 10), se realiza una nueva ejecución de este para determinar la

capacidad que tiene para discernir entre ambos casos. En la tabla 4.10 se muestran los resultados obtenidos. BLAST logra discernir entre enzimas y no enzimas con un muy alto rendimiento, lo que sugiere la posibilidad de incorporar este filtro como primer paso ante cualquier pipeline que se proponga. El punto más bajo está relacionado con los falsos negativos de los casos en que no es enzima; es decir, a BLAST le cuesta reconocer cuando una proteína no es una enzima. Sin embargo, dado que los puntajes asociados a las enzimas son muy altos, esos falsos negativos se traducen en secuencias no alineadas en su mayoría, puesto que si BLAST a esos casos los asignara como enzimas, la precisión de estas últimas disminuiría al tener más falsos positivos. Dado que el interés de este trabajo es mejorar la clasificación de enzimas, lo que importa es poder reconocerlas, independiente de lo que pase con las que no lo son.

	Precision	**Recall**	**F1**	**Cantidad**
Enzimas	99.16 %	99.44 %	99.30 %	188.179
No enzimas	99.80 %	92.96 %	96.26 %	242.954

Tabla 4.10: Rendimiento de BLAST al separar enzimas de las que no lo son.

4.3. Propuesta

Las pruebas realizadas muestran que la alineación de secuencias sigue siendo un buen acercamiento para un gran conjunto de clases. Por otro lado, el uso de InterPro como descriptor de enzimas muestra resultados promisorios si se piensa en generalizar el modelo, lo que reafirma lo propuesto en [4] en donde se afirma que la información extraída de InterPro corresponden a los descriptores más influyentes cuando se quiere clasificar funcionalidad, mientras que la similitud de secuencias es el segundo (fig 4.3) . Si a esto se le agrega que los métodos que han tenido los mejores resultados en CAFA incorporan fuentes heterogéneas de información, es natural pensar en un método que ensamble ambos acercamientos probados. Es por ello que se propone BLAST-KNN: un ensamble entre BLAST (la implementación de *Diamond*) y un algoritmo clasificador que venga a complementar las clases en que el primero no tenga un rendimiento perfecto. De esta manera, se reduce el problema de clasificación a subconjuntos de clases previamente estudiados, optimizando la búsqueda y la posibilidad de ajustar parámetros a cada caso. Dado que las clases poco representadas son igual de importantes que las ampliamente muestreadas, se utiliza la variación macro de la norma F1.

Los pasos para implementar BLAST-KNN son :

1. Obtención de conjunto de enzimas de entrenamiento : Se obtiene la información contenida en SwissProt. Se realiza el mismo procedimiento de filtrado previamente mostrado: filtro de redundancias usando *CD-HIT*, extracción enzimas que contengan los 4 dígitos EC y que la clase a la que pertenecen tenga al menos dos muestras. Para los casos de clases que tengan más de un número EC asociado, se consideran como una clase independiente. Es decir, si una enzima pertenece a la clase A y otra a la A y B a la vez, se considera que pertenecen a clases distintas.

2. Búsqueda por similitud: Se ejecuta BLAST sobre el conjunto de datos. La base de datos de referencia de donde se extrae la secuencia más similar corresponde al mismo conjunto de datos a evaluar,

Descriptors (number)	AC50 ± standard error of the mean	
	only descriptors from col. 1	excluding descriptors from col. 1
All (24)	0.96 ± 0.02	–
Optscore (1)	0.87 ± 0.03 [a]	0.96 ± 0.03
Sequence similarity (4)	0.88 ± 0.04	0.95 ± 0.03*
Sequence length attributes (3)	0.69 ± 0.06	0.96 ± 0.03
Synteny scores (1)	0.64 ± 0.07 [a]	0.96 ± 0.03
Alignment free sequence similarity (1)	0.70 ± 0.06 [a]	0.96 ± 0.03
InterPro domain (8)	0.88 ± 0.04	0.92 ± 0.03*
PEDANT3 properties (7)	0.49 ± 0.07	0.99 ± 0.03
Sequence similarity + InterPro domain (12)	0.95 ± 0.03	0.85 ± 0.05
PEDANT3 properties + InterPro domain (15)	0.93 ± 0.07	0.91 ± 0.05*

[a] No machine learning was used; *significant difference at $P < 0.01$ according to the bootstrap test with 10 000 replicas.

Figura 4.3: Rendimiento según diferentes grupos de descriptores, usando una red neuronal asociativa (ASNN) sobre un conjunto de cuatro genomas de bacterias. Se extraen de a uno los descriptores mostrados para verificar el peso que tienen en la clasificación. La red neuronal asociativa está diseñada por los mismos autores del trabajo. Fuente e imagen extraídos de [4]

en donde evidentemente no se considera la alineación consigo mismo como válida. Con ello se obtiene una estadística de las clases en que la búsqueda por similitud es perfecta y en las que es posible mejorar. Para efectos prácticos, se define la clase $*$ como aquella que engloba secuencias no alineadas. Se extraen aquellas con un puntaje F1 <1.0, que generan un conjunto X. Se define la siguiente condición:

- $c_a(b)$ = Será verdadera cuando exista al menos una enzima $z \in b$ que se predijo que pertenecía a a siendo que pertenece a b. Esto es un falso negativo para la clase b y falso positivo para a.

La condición permite definir el conjunto C_{x_i}, $\forall x_\mathrm{i} \in X$ como

$$C_{x_\mathrm{i}} = \left\{ x_j \in X / c_{x_\mathrm{i}}(x_j) \vee c_{x_j}(x_\mathrm{i}) \right\} \tag{4.1}$$

Es así como se obtiene $|X|$ (| | representa la cardinalidad del conjunto) conjuntos en que cada uno representa las clases con que se equivoca x_i $\forall$i. Para los casos en que $|C_{x_\mathrm{i}}| = 1$ y esa única clase es $*$, se tiene una clase en que no se equivoca con ninguna otra, tan solo el BLAST no puede alinear alguna de sus muestras. Dado que no se tiene con que clase comparar, esos casos se apartan del conjunto X. Se define entonces :

$$X' = X \backslash Y \tag{4.2}$$

Siendo $Y = \{x_\mathrm{i} \in X / |C_{x_\mathrm{i}}| = 1 \wedge C_{x_\mathrm{i}} = *\}$ y $\backslash$ la diferencia entre ambos conjuntos. Se guarda en un diccionario los puntajes F1 obtenidos en cada clase en X'.

3. Para cada C_{x_j} con $x_j \in X'$, se entrena un clasificador. Si su puntaje F1 macro sobre x_j es mayor al obtenido por BLAST, entonces el clasificador logra identificar de mejor manera los casos que pertenecen o no a dicha clase. Se genera un diccionario de clasificadores con sus puntajes sobre la clase respectiva.
   El modelo propuesto es similar al presentado en la fig 4.1 pero se debe ajustar puesto que muchas de las clases de interés tienen muy pocos representantes (hasta dos muestras). Una opción tomada por la mayoría de los trabajos ([10], [15] entre varios) es definir un mínimo de muestras por clase y no trabajar con las menores a esa cantidad. Esto les permite usar acercamientos de *machine learning* como SVM o Random Forest o redes neuronales. Sin embargo, es del interés del proyecto llegar a clasificar esas clases con bajos representantes, por lo que se decide utilizar un algoritmo más simple pero a la vez poderoso como KNN, el cuál ha mostrado muy buenos resultados en diversos trabajos como [11] o [3]. En resumen, los cambios implementados corresponden a :

   (a) En vez de separar un conjunto de entrenamiento y validación, se realiza un validación cruzada anidada. Los ciclos interno y externo se escogen de acuerdo a la cantidad máxima de muestras por clase. Si la cantidad de muestras máximas por clase es mayor a 10, entonces:
   - Ciclo interno : Stratified 5-fold
   - Ciclo externo: Stratified 10-fold

   Caso contrario:
   - Ciclo interno : LeaveOneOut
   - Ciclo externo : LeaveOneOut

   De esta manera, en conjuntos de clases con pocas muestras se prefiere sacar de a una puesto que son pocos. Para conjuntos más grandes, se realizan particiones de a 5 y 10. En el ciclo interno se buscan los parámetros óptimos del algoritmo mientras que en el externo se validan con datos no conocidos. El umbral para escoger alguno de los dos casos (10) se escogió heurísticamente. El caso LeaveOneOut solo se calcula una vez, puesto que los resultados son los mismos; en el otro caso se realizan 3 corridas de la validación cruzada, obteniendo 3 posibles modelos, escogiendo los parámetros que más se repitan.

   (b) Uso de descriptores de InterPro en vez de ProFet. Esto es lo que le da fuerza al modelo, teniendo una representación compacta e informativa de cada enzima. Se define un kernel $K : A \to B$ en donde A es el espacio de las enzimas representadas por su secuencia de largo variable y B es el espacio de su representación en base a descriptores de InterPro. Esto se aplica para cada C_{x_j} de manera independiente:

      i. Uso de InterProScan sobre todas las enzimas de las clases en C_{x_j}, usando todas las bases de datos que ofrece InterPro.

      ii. Se obtiene la cantidad distintas de descriptores (n) existente en todo el conjunto previamente analizado. Este número dará el largo del vector con que se describe cada enzima y la clase en sí. Cada columna representa un descriptor distinto (ordenados alfabéticamente).

      iii. Por cada enzima se recorren las columnas, colocando un 1 o 0 dependiendo si ese descriptor está asociado a ella o no. El kernel mapea a un espacio binario $\{0, 1\}^n$, siendo n

propio de cada C_{x_j}.
La unión de todo los vectores (por cada enzima) genera una matriz de dimensiones MxN con M = Cantidad total de enzimas en C_{x_j} y N = Cantidad total de descriptores distintos generados en las enzimas de C_{x_j}. Como se ve, por cada conjunto C_{x_j} lo valores de M y N variarán, pero dado que para cada caso se entrena un clasificador, esto no es problema.

Para ejemplificar lo anteriormente mencionado, se muestra el siguiente caso. Q86BA1, Q8TDZ2 y E9QYP0 son identificadores para enzimas en SwissProt. Las dos primeras pertenecen a la misma clase (1.14.13.225) y la tercera a una distinta (1.14.13.196) Al usar InterProScan para obtener sus descriptores, se obtiene:

- Q86BA1 : PF00307, PF12130, PF00412, SM00132
- Q8TDZ2 : PF00307, PF12130, PF00412, PF01494, SM00033, SM00132
- E9QYP0 : PF13434

Para este caso, el cual corresponde a un ejemplo, se buscó solo en Pfam y SMART. Existen $n = 7$ distintos descriptores (PF00307, PF00412, PF01494, PF12130, PF13434, SM00033, SM00132), por lo que todas las enzimas de este grupo serán representadas por un vector binario de largo 7.

- Q86BA1 =$\{1, 1, 0, 1, 0, 0, 1\}$
- Q8TDZ2 = $\{1, 1, 1, 1, 0, 1, 1\}$
- E9QYP0 = $\{0, 0, 0, 0, 1, 0, 0\}$

Esto genera una matriz de 3×7 la cual será la entrada al algoritmo de *machine learning*.

$$M = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

(c) Dado que los vectores son binarios, ya no se requiere normalizar por lo que se remueve el uso del estandarizador (standardscaler).

(d) Se escoge KNN como clasificador dado los resultados previos obtenidos.

El modelo se muestra en la fig 4.4. Se almacena en un diccionario los puntajes obtenidos por cada clasificador.

Una vez realizado el proceso de implementación, se obtiene un conjunto de clasificadores. Con ello, el algoritmo ensamble comprende el uso de BLAST y los clasificadores, usando estos últimos en los casos en que hayan obtenido un puntaje F1 mayor que BLAST según lo visto en la implementación. Las etapas que tiene el algoritmo son:

1. Usando la secuencia de cada enzima, se usa BLAST para asignarle una clase.
2. Dada la clase asignada por BLAST, se busca el puntaje obtenido en esa clase tanto por BLAST como por el clasificador en el período de implementación.

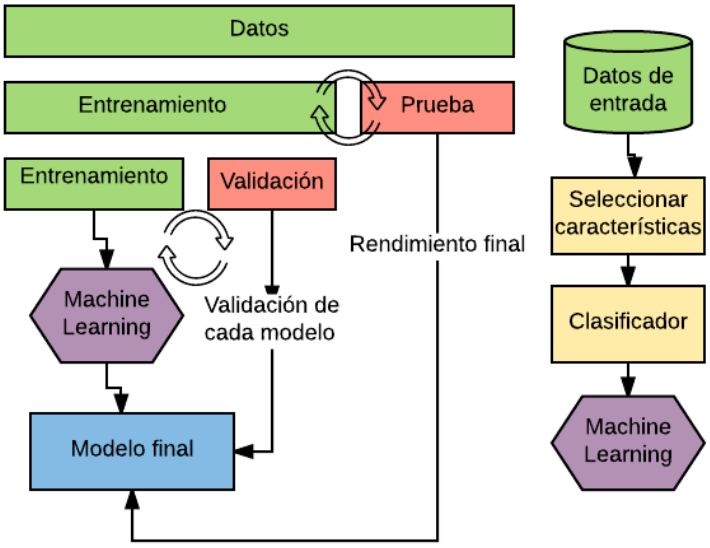


Figura 4.4: Diagrama de flujo. Se representa el proceso de entrenamiento del conjunto de clasificadores que conforman el algoritmo propuesto. La parte derecha muestra en detalle lo que ocurre desde que se obtienen los datos de entrenamiento hasta que se ingresan al clasificador.

3. Si BLAST había obtenido un mejor puntaje, entonces se asigna la clase misma. De lo contrario, se usa el clasificador asociado a esa clase para asignarle una etiqueta. Para ello se debe usar InterProScan para obtener los descriptores y transformarlo al vector de largo fijo que representa a la clase. Con ello se puede realizar la predicción.

El procedimiento se muestra en la fig 4.5 a modo de diagrama de flujo.

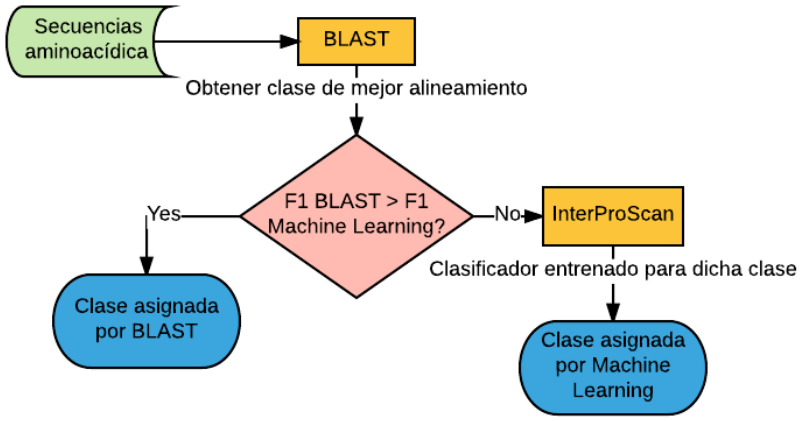


Figura 4.5: Diagrama de flujo de algoritmo de ensamble.

Capítulo 5

Resultados

En esta sección se procede a mostrar las pruebas realizadas al algoritmo ensamble propuesto BLAST-KNN, así como los resultados obtenidos. Primero se muestra su efectividad usando parámetros por defecto. Posteriormente se ajustan los parámetros de KNN y del seleccionador de características escogido. Para este último se muestran 2 casos: F-Test e información mutua. Se realiza una comparación entre el uso de todos los descriptores que proporciona InterPro y el solo uso de sus descriptores propios. Finalmente, se realiza una prueba sobre actualizaciones recientes de SwissProt. Todo esto comparado con la alternativa de solo usar BLAST.

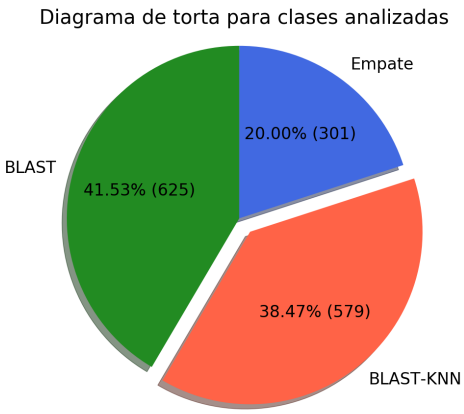
5.1. Modelo sin ajuste de parámetros

Los parámetros por defecto corresponden a $K = 1$ y el uso del total de las características. Se implementa el modelo según lo pasos mostrados en el Capítulo 4. En la fig 5.1a se muestra que hay 579 clases (38.47 %) que fueron mejoradas utilizando el modelo propuesto, es decir, el puntaje F1 obtenido por el algoritmo propuesto sobrepasa al obtenido solo por BLAST. En 625 (41.53 %) BLAST sigue siendo una mejor alternativa y empatan en 301 clases. Para los casos multiclase, el modelo propuesto presenta mejores resultados. En la tabla 5.1 se muestra el porcentaje promedio en que se logra mejorar el puntaje F1 de las clases. Esto se calcula como la diferencia entre los valores F1 obtenidos por una misma clase con ambos acercamientos. Se promedia todo y se muestra su desviación.

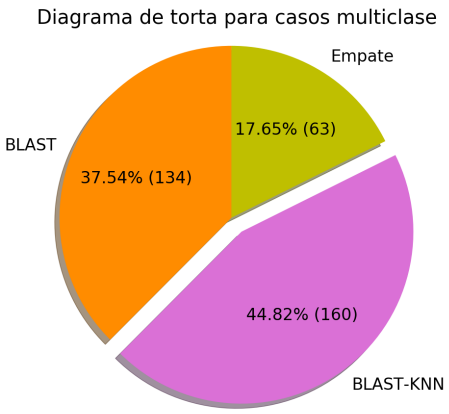
Se ve una alta desviación lo que indica que existen clases en donde uno de los acercamientos gana considerablemente. Esto se ve en clases en donde se tiene un 0 % de aciertos (ya sea BLAST o el modelo)

Porcentaje de mejora de métrica F1 en clases ganadas por BLAST	**Porcentaje de mejora de métrica F1 en clases ganadas por BLAST-KNN**
18.16 / 23.11 %	10.89 / 17.51 %

Tabla 5.1: Promedio y desviación en que cada acercamiento supera al otro en términos de diferencia entre sus puntajes F1. Caso sin ajuste de parámetros.



(a) Porcentaje de clases mejoradas usando modelo propuesto sin ajuste de parámetros.



(b) Comportamiento en casos multiclase cuando se usan parámetros por defecto.

Figura 5.1: Distribución de clases mejoradas por uno u otro acercamiento al usar los descriptores proporcionados por InterPro. Se usan los parámetros por defecto en BLAST-KNN.

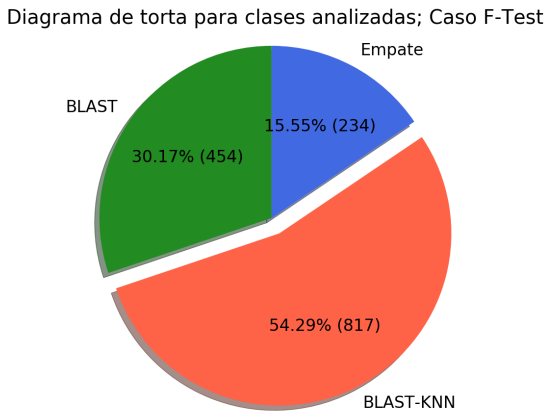
y un 100 % por parte del contrario. Esto reafirma la importancia de usar un modelo de ensamble, pues ambos se complementan en casos donde el otro falla.

5.2. Modelo con ajuste de parámetros

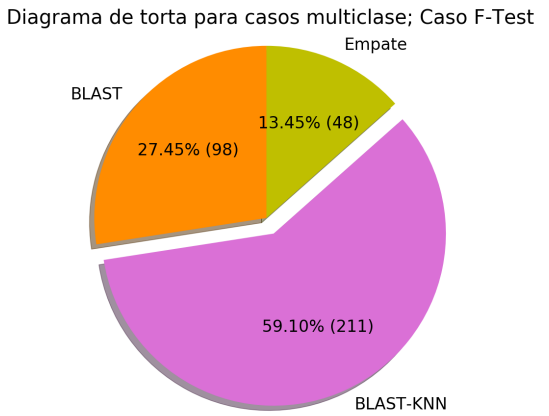
Usando la clase **GridSearchCV** de *scikit-learn* y la validación cruzada anidada propuesta anteriormente, se realiza una búsqueda exhaustiva de los parámetros del modelo: el número de vecinos cercanos a considerar en KNN y el número de características relevantes a utilizar. Para obtener este último, se usan dos evaluaciones distintas en que uno busca una dependencia lineal y el otro una no lineal, entre características y etiquetas.

5.2.1. Caso F-Test

Los diagramas de torta en la fig 5.2 dan muestra del considerable aumento de rendimiento del modelo ante el ajuste de sus parámetros. En 5.2a se ve que más de la mitad de las clases son clasificadas de mejor manera por el modelo propuesto, frente al 38.4 % obtenido anteriormente. De paso se disminuyen las clases en que ambos empatan. En 5.2b se reafirma la mejor performance del modelo frente a casos multiclase, obteniendo cerca de un 60 % de victorias. La efectividad radica en el uso de un selector de características, puesto que KNN solo se basa en distancias, sin dar prioridad a las características usadas. Con el selector, solo se considera distancia en las que realmente aportan información, dejando fuera los datos irrelevantes, aumentado así considerablemente el rendimiento.



(a) Porcentaje de clases mejoradas usando modelo propuesto con ajuste de parámetros. Se seleccionan características por medio de F-test.

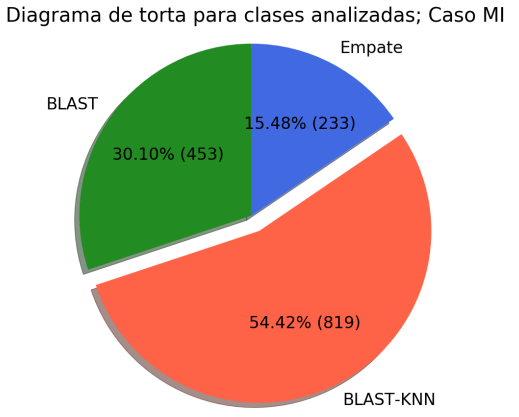


(b) Comportamiento en casos multiclase para modelo con ajuste de parámetros. Se seleccionan características por medio de F-test

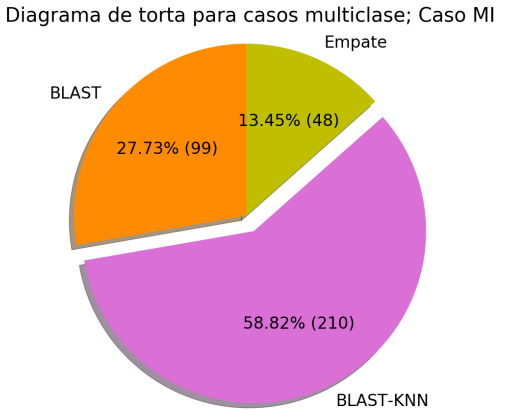
Figura 5.2: Distribución de clases mejoradas por uno u otro acercamiento al usar los descriptores proporcionados por InterPro. Se seleccionan las características relevantes por medio de F-test.

5.2.2. Caso Información mutua (MI)

Los resultados son casi idénticos al caso anterior. En 5.9a se muestra que el modelo logra clasificar de mejor manera dos clases adicionales, mientras que en 5.9b se pierde una.



(a) Porcentaje de clases mejoradas usando modelo propuesto con ajuste de parámetros. Se seleccionan característica por medio de información mutua.



(b) Comportamiento en casos muticlase para modelo propuesto con ajuste de parámetros. Se seleccionan características por medio de información mutua.

Figura 5.3: Distribución de clases mejoradas por uno u otro acercamiento al usar los descriptores proporcionados por InterPro. Se seleccionan las características relevantes por medio de información mutua.

5.2.3. Porcentajes de mejora sobre clases

De acuerdo con lo mostrado en la tabla 5.2, se ve que en comparación a la tabla 5.1, los porcentajes con que BLAST y BLAST-KNN ganan se estabilizan. Esto indica que el algoritmo propuesto aumentó su rendimiento, logrando mejorar más el puntaje en las clases donde vence, y de paso disminuir la brecha en las clases en que BLAST logra un mejor resultado. La existencia de altas desviaciones sigue indicando los casos de clases en donde la mejora por uno de los acercamientos es radical.
Si se compara la tabla 5.2 con la tabla 5.1, se puede observar que al usar información mutua como seleccionador de características se logra mejorar el puntaje F1 con que se mejoran las clases (13.45 % versus el 10.89 % que se obtenía sin ajustar parámetros).

Caso	Porcentaje de mejora de métrica F1 en clases ganadas por BLAST	Porcentaje de mejora de métrica F1 en clases ganadas por BLAST-KNN
F-Test	14.08 /19.94 %	13.05 / 19.68 %
MI	14.12 / 19.97 %	13.45 / 20.19 %

Tabla 5.2: Promedio y desviación en que cada acercamiento supera al otro en términos de diferencia entre sus puntajes F1. Caso con ajuste de parámetros y diferentes seleccionadores de características.

5.2.4. Número de vecinos utilizados en KNN

En la fig 5.4 se muestra la distribución de vecinos óptimos utilizados por cada caso sobre el total de clases, considerando las clases empatadas y superadas por BLAST-KNN. Se ve que para el caso F-Test, existe una mayor cantidad de clases que utiliza 1 vecino, mientras que para el caso de información mutua supera en el uso de 5 vecinos. Sin embargo, la distribución obtenida para ambos casos no difiere mayormente en relación a la cantidad de veces que se ocupa cada cual.

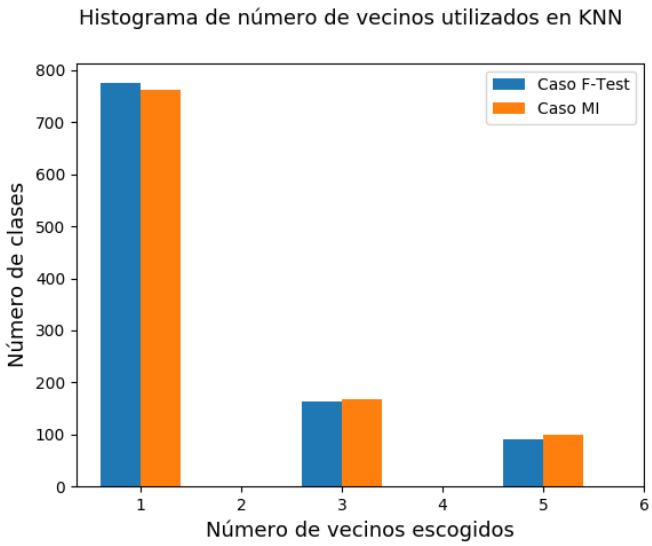


Figura 5.4: Distribución de número de vecinos óptimos usados por el algoritmo propuesto ante los distintos seleccionadores de características probados.

5.2.5. Número de características seleccionadas

En la tabla 5.3 se muestra la distribución del número de características seleccionadas en los casos en que se supera o empata el puntaje F1 obtenido por BLAST. Se ve una predominancia en escoger menos de 15 de ellas, siendo un poco más del 75 % de los casos. La disminución es gradual, lo que muestra la existencia de características de gran impacto que no necesitan estar en gran cantidad para permitir discriminar. Los casos que escogen más de 50 puede que estén con mucha redundancia, es decir, muchas características igual de importantes pero que dicen lo mismo. Este problema es uno de los defectos de los acercamientos *univariate* dentro de los métodos de filtrado. Sin embargo, representan menos del 8 % de los casos, los que pueden ser estudiados con mayor detalle más adelante.

Rangos de características	**Número de clases**		**% del total**	
	F-Test	MI	F-Test	MI
<15	815	810	77.54 %	76.99 %
[15,50[	153	150	14.55 %	14.25 %
[50, 100[	41	46	3.90 %	4.37 %
[100,500[	29	34	2.75 %	3.23 %
[500,4000[	13	12	1.23 %	1.14 %

Tabla 5.3: Distribución de número de características escogidas por clase ante distintos seleccionadores de características. Caso en que se utilizan los descriptores entregados por InterPro.

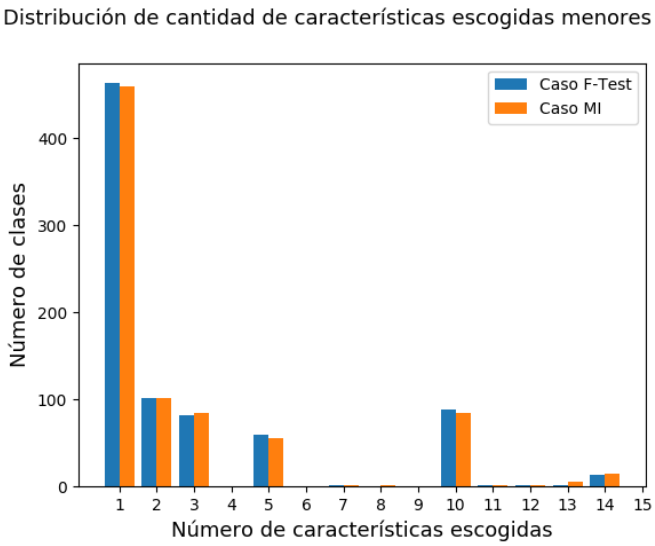
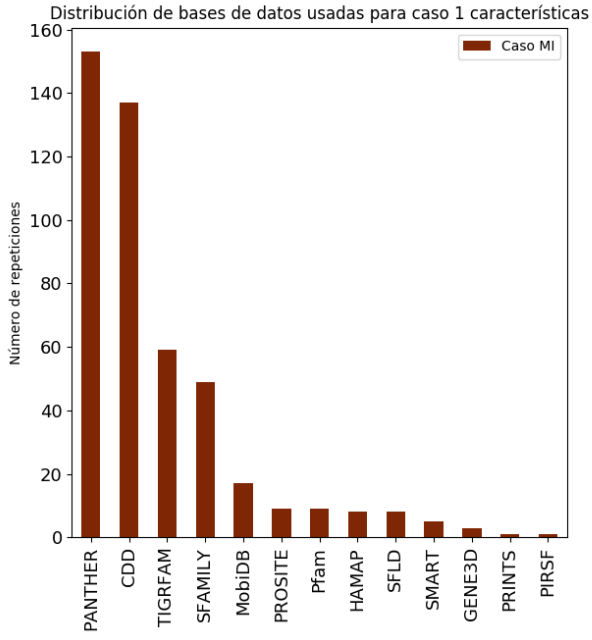


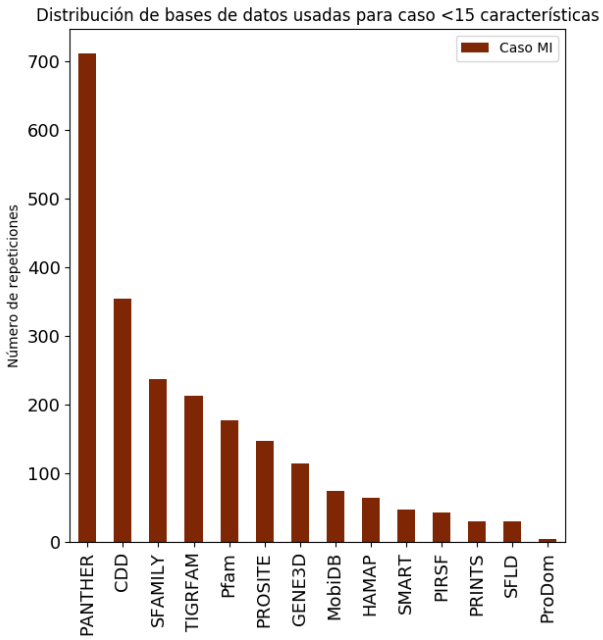
Figura 5.5: Acercamiento a tabla 5.3 para el caso más representado (menos de 15 características escogidas). Se ve un predominio de los casos en que se escoge solo una característica para separar clases.

En la fig 5.5 se realiza un acercamiento al primer caso de la tabla (el más representado) para entender su distribución. Se ve que en mucho de los casos se escoge una característica como la más relevante con la que basta para discernir entre clases.

Dado que el uso de información mutua entrega resultados levemente mejores, se escoge para estudiar cuáles son las bases de datos más usadas. Esto se muestra en los histogramas de la fig 5.6, en donde 5.6a muestra la base más recurrida en casos en que se escoge solo una característica, mientras que en 5.6b se muestra lo mismo pero entre los casos en que se escogen menos de 15 características. PANTHER es la base con mayor influencia en ambos casos, probablemente dado que la manera en que ellos asignan familias de enzimas es considerando su función molecular, además de contar con revisión manual. CDD presenta un rendimiento similar, mostrando que el uso de dominios altamente conservados evolucionariamente son prueba de una funcionalidad común. TIGRFAM y SUPERFAMILY se reparten el tercer y cuarto lugar de relevancia. En particular, la segunda realiza una clasificación estructural, lo que muestra la relevancia de la estructura de una enzima con respecto a su función. Se hubiese esperado que HAMAP tuviese mayor relevancia dado su anotación manual y uso para clasificar datos de TrEMBL.
Los casos sin barras se dan por el espacio de búsqueda que se usó para encontrar el mejor número de características, el cual consideraba valores aleatorios dentro del intervalo [1, total de características], por lo que algunos valores (4,9,11) no fueron considerados.
Se escoge el uso de información mutua como parte del modelo dado los resultados levemente mejores y la posibilidad de extraer cualquier tipo de relación entre los descriptores y las clases.



(a) Histograma de bases de datos usadas para el caso en que se elige solo una característica para separar clases.

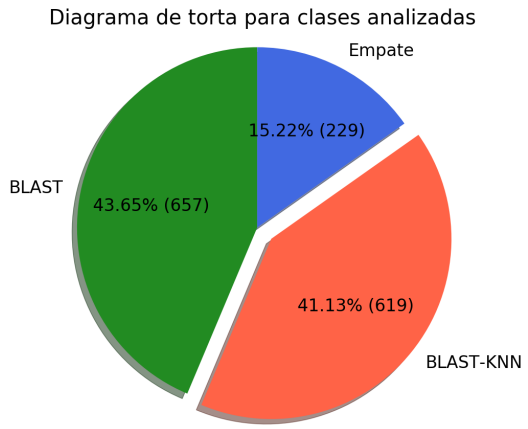


(b) Histograma de bases de datos usadas para el caso en que se escogen menos de quince características para separar clases.

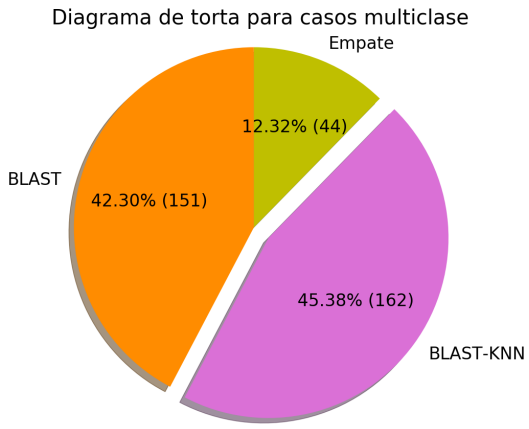
Figura 5.6: Distribución de bases de datos de InterPro más usadas luego de la selección de características usando información mutua. Se observa que los descriptores obtenidos de PANTHER son los más usados.

5.3. Diferencias con IPR

En [11] se hace referencia a la relevancia de los descriptores propios que provee InterPro. Se realizan las mismas pruebas anteriormente mostradas (ajustando parámetros y usando información mutua como selector de características) pero esta vez representando cada enzima por los descriptores provistos por InterPro. En la fig 5.7 se muestra la distribución de clases mejoradas, empatadas y perdidas. Se ve que el uso de estos descriptores mejora levemente el caso sin ajuste de parámetros, dejando en claro que el uso de los descriptores de cada base de datos es más informativo y provee mayor diversidad de fuentes. A pesar de que los descriptores de InterPro se extraen agrupando proteínas que comparten ciertas clasificaciones en las distintas bases de datos, no queda claro cuál es el criterio que se utiliza, pudiendo darse el caso en que al agrupar información y tratarlos a todos como igual se pierda el nivel de especificidad necesario para clasificar al 4to EC. Sin embargo, al revisar la tabla 5.4, esta muestra que con esta configuración, se logra aumentar la brecha con que se mejoran las clases con respecto a lo obtenido en la tabla 5.2, lo que sugiere que el uso de estos descriptores podría mejorar lo obtenido en 5.3. En resumen, hay una menor cantidad de clases mejoradas, pero en promedio el porcentaje en que estas aumentan es mayor a lo obtenido.



(a) Porcentaje de clases mejoradas usando modelo propuesto con ajuste de parámetros. Las características corresponden a los descriptores IPR.



(b) Comportamiento en casos multiclase para modelo con ajuste de parámetros, usando descriptores IPR e información mutua para seleccionar.

Figura 5.7: Distribución de clases mejoradas por uno u otro acercamiento al usar solos los descriptores IPR. Se seleccionan las características relevantes por medio de información mutua.

5.4. Mezcla de características

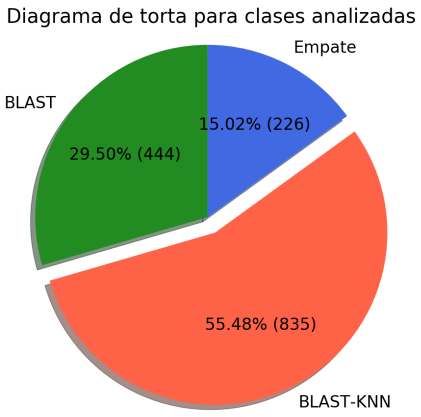
Dado lo planteado en la sección anterior, se realiza una prueba en que cada enzima está descrita usando tanto los descriptores de cada base de datos como los propios de InterPro. Esto eleva el número de características existentes. Se realiza una búsqueda de parámetros y se usa información mutua para extraer

Porcentaje de mejora de métrica F1 en clases ganadas por BLAST	Porcentaje de mejora de métrica F1 en clases ganadas por BLAST-KNN
26.98 / 29.03 %	14.26 / 27.71 %

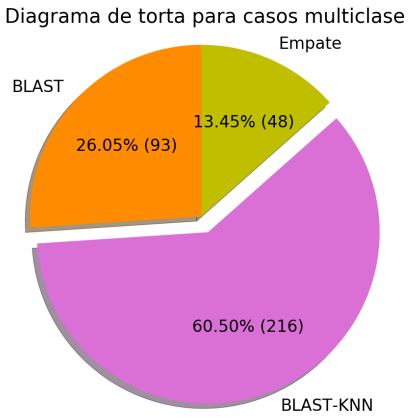
Tabla 5.4: Promedio y desviación en que cada acercamiento supera al otro en términos de diferencia entre sus puntajes F1. Caso en que se utilizan solo los descriptores IPR, junto a un ajuste de parámetros y selección de características vía información mutua.

las características más informativas.

En la fig 5.8 se muestra que la combinación tuvo un efecto positivo, pues se logran mejorar 16 nuevas clases, además de acrecentar la ventaja de este modelo frente a los casos multiclases. La tabla 5.5 muestra que el porcentaje de mejora de las clases es similar al obtenido en el caso 5.2, por lo que la ventaja de mezclar descriptores se refleja en un aumento de clases mejoradas, no así en el porcentaje en que se mejora el puntaje F1 en las respectivas clases.



(a) Porcentaje de clases mejoradas usando modelo propuesto con ajuste de parámetros. Las características corresponden a una mezcla entre los descriptores de las distintas bases de datos y los descriptores IPR.



(b) Comportamiento en casos multiclase para modelo con ajuste de parámetros, usando una mezcla de descriptores.

Figura 5.8: Distribución de clases mejoradas por uno u otro acercamiento al usar los descriptores IPR junto a los proporcionados por InterPro. Se seleccionan las características relevantes por medio de información mutua.

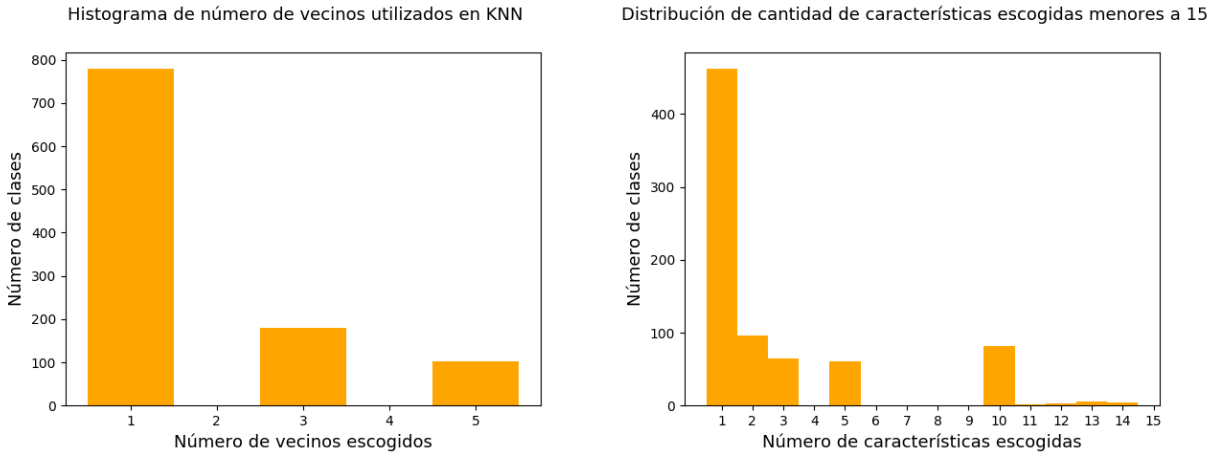
Porcentaje de mejora de métrica F1 en clases ganadas por BLAST	Porcentaje de mejora de métrica F1 en clases ganadas por BLAST-KNN
14.57 / 20.29 %	13.41 / 20.12 %

Tabla 5.5: Promedio y desviación en que cada acercamiento vence al otro en términos de diferencia entre sus puntajes F1. Caso en que se utilizan una mezcla de descriptores: los descriptores IPR y todas las bases de datos disponibles en InterPro, junto a un ajuste de parámetros y selección de características vía información mutua.

La distribución del número de características escogidas (tabla 5.6), así como la del número de vecinos escogidos y la distribución de de características escogidas cuando se escogen menos de 15 (fig 5.9) no muestra variaciones significativas.

Rango de características	Números de clases	% del total
<15	777	73.23 %
[15,50[	155	14.61 %
[50,100[	59	5.56 %
[100,500[	53	4.99 %
[500,4000[	17	1.60 %

Tabla 5.6: Distribución de número de características escogidas por clase. Caso en que se utilizan los descriptores obtenidos de las distintas bases de datos de InterPro junto a los descriptores IPR.



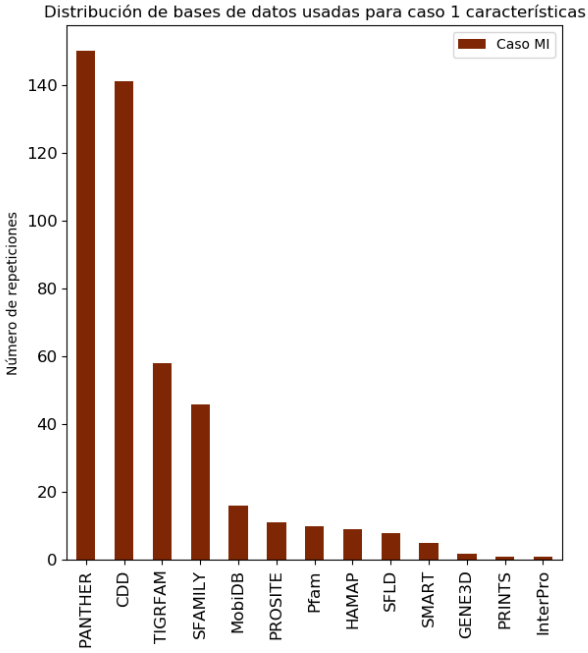
(a) Distribución de número de vecinos óptimos.

(b) Casos que escogen menos de 15 características.

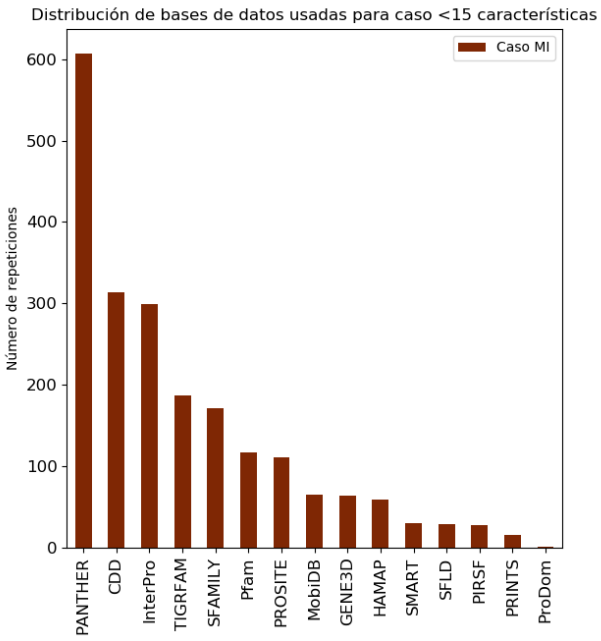
Figura 5.9: Acercamiento a tabla 5.6 para el caso más representado

Sin embargo, en lo que sí se nota cambios es en las bases de datos escogidas, puesto que los descriptores de InterPro (IPR) toman fuerza cuando se escogen menos de 15 características, llegando al tercer

lugar de relevancia (fig 5.10b), muy cerca de CDD. Esto dice que, si bien por sí solo no son capaces de discriminar entre clases, puede que dada la compactación de información que contienen, sirva como primera etapa para agrupar enzimas de la misma funcionalidad, pero luego requiere información extra que entregue el nivel de especificidad para discriminar.



(a) Histograma de bases de datos usadas para el caso en que se elige solo una característica para separar clases. InterPro hace referencia a los descriptores IPR.



(b) Histograma de bases de datos usadas para el caso en que se escogen menos de quince características para separar clases. InterPro hace referencia a los descriptores IPR.

Figura 5.10: Distribución de bases de datos de InterPro más usadas luego de la selección de características usando información mutua, al juntar descriptores de las distintas bases de InterPro junto a los descriptores IPR. Se observa la relevancia que toman los descriptores IPR.

Dado estos resultados, se modifica levemente la propuesta original, puesto que para describir las enzimas se considera una mezcla entre los descriptores de las distintas bases de datos y los propuestos por InterPro.

5.5. Pruebas a nuevos datos

5.5.1. Actualizaciones de SwissProt

Se obtienen las actualizaciones de SwissProt de junio y julio (release 2017_06 y 2017_07) con el fin de probar el acercamiento ante estos nuevos datos. El conjunto cuenta con 600 nuevas proteínas, de las cuales 152 corresponden a enzimas con todos sus dígitos EC repartidas en 74 clases. Al aplicar BLAST, este se equivoca en 40 de las 74 clases. Al momento de probar el modelo propuesto, 39 de las clases

no se pueden analizar puesto que o son clases nuevas, o BLAST había registrado un rendimiento del 100 %, por lo que no se habían considerado. En el caso de la clase que teóricamente se puede analizar, se encuentra con que no se tenía registro que BLAST se confundía entre dicha clase real y la clase que predijo. Por lo tanto no se pudo analizar los nuevos datos incorporados, revelando que el modelo requiere actualizaciones constantes en la medida que la base de datos de referencia se va incrementando.

Capítulo 6

Discusión

6.1. Secuencias no alineadas por BLAST

La primera etapa del ensamble propuesto se sustenta en BLAST, por lo que hereda sus limitantes. Una de ellas corresponde a los casos en que no es capaz de alinear ninguna secuencia con la base de datos usada. Una posible solución es incorporar una base de datos con más variedad de secuencias, como lo es TrEMBL. Para probar lo anteriormente dicho, se recopilan las secuencias no alineadas en el experimento realizado en 4.2, en el caso en que se usó un e-value igual a 10 y la configuración *more sensitive*, la cual fue la escogida para el modelo. Se realiza una nueva corrida de BLAST usando TrEMBL como fuente de búsqueda de alineaciones y la misma configuración usada en el experimento original. Solo se consideran alineaciones válidas como aquellas que tengan un e-value menor a 1.

De los 319 casos no alineados en la sección 4.2, 218 de ellas sí se alinean con esta nueva base de datos. Si se analiza el resto que no fue alineado, se ve que la mayoría tiene un largo bastante reducido por lo que se tiene la duda si estos fragmentos realmente corresponden a información válida. En futuras actualizaciones de UnitProtKB se podrían modificar o remover. El histograma de largos en las secuencias no alineadas se muestra en la figura 6.2. Esto da cuenta de que en la medida que la información de TrEMBL se revise manualmente y se pase a SwissProt, se podrán alinear secuencias que hoy en día no lo hacen, y de esta manera incorporarlos al ensamble.

6.2. Ventajas de BLAST-KNN en casos multiclase

Como se vio en la fig. 5.8b y en los resultados previamente obtenidos, BLAST-KNN tiene un buen rendimiento en las clases que tienen más de un número EC asociado. Esto se refleja en una alta cantidad (60.05 %) de clases con más de un EC mejoradas por el algoritmo propuesto. Parte de estos casos corresponden a situaciones en que se comparan clases que comparten algunos de sus números EC. Por ejemplo, la clase 1.3.1.76 - 4.99.1.4 indica que las enzimas que pertenezcan a dicha clase pueden tener ambos EC. A su vez, existe la clase 2.1.1.107-1.3.1.76-4.99.1.4, la cual indica que los miembros de dicha

clase pueden realizar cualquiera de las tres funciones asociadas a los tres números EC. Se ve que ambas clases comparten dos de los tres números EC. Al evaluar con BLAST, este se equivoca en asignar la clase correcta en algunas enzimas. Es el caso de la enzima O14172 perteneciente a SwissProt, a la cual BLAST le asigno la clase con tres números EC's, siendo que su etiqueta corresponde a la clase con solo dos EC's. El origen de esta clasificación errónea se ilustra en la fig 6.1. La barra negra indica la enzima evaluada (O14172) y la barra verde indica el mejor alineamiento encontrado por BLAST en SwissProt (enzima B8GUD3 perteneciente a la clase de tres EC's). Con colores más remarcados y encerrados en un rectángulo se encuentran las partes de las secuencias que comparten ambas enzimas, lo que correspondería a dos de las 3 funcionalidad comunes. Sin embargo, se puede ver que existe una gran parte que no comparten. Es ahí donde se produce la diferencia entre ambas, puesto que la enzima correspondiente a la barra verde realiza una tercera función, asociada a algún fragmento de la parte no alineada. Dado que es el mejor alineamiento encontrado, se le asigna la clase de tres dígitos EC's a la enzima 014172, a pesar de no tener la parte de la secuencia asociada al número EC 2.1.1.107.
Es en esos casos en donde BLAST-KNN, gracias a los descriptores extraídos de InterPro, logra identificar diferencias entre ambas clases a pesar de compartir dos de las tres funcionalidades. En este caso particular, el descriptor cd11642, proveniente de la base de datos CDD, esta asociado a las enzimas que pueden realizar las tres funcionalidades, pero no está asociado a las que solo pueden realizar dos, logrando separar fácilmente entre ambas clases.

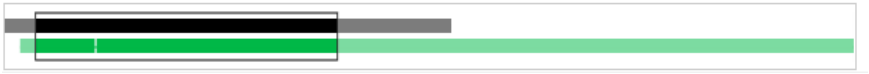


Figura 6.1: Alineación de enzima 014172 con enzima B8GUD3 (la secuencia más parecida en SwissProt).

Al presentarse estos casos en que se comparten números EC's pero a la vez se difiere en alguno, queda la duda si realmente existen estos dos grupos de enzimas por separado, o simplemente aquellas que les falta algún número EC es debido a un problema en el ensamble de la información obtenida de los genes. Hay que destacar que no siempre se trabaja con genomas, los cuales contienen toda la información genética, sino que también con transcriptomas, los cuales muestran una parte de la información codificada en los genes. Podría darse que haya faltado información que codificar, lo que se traduce en que faltó ensamblar una parte de la secuencia aminoacídica. Esto se podría resolver mediante conocimiento experto, analizando los codones de parada que indican si la codificación de donde viene la secuencia es correcta o está recortada.

6.3. Limitación de SwissProt

SwissProt tiene la ventaja de ser información manualmente curada, disminuyendo la posibilidad de tener etiquetas mal asignadas que disminuyan el rendimiento del modelo. Sin embargo, la limitante que trae tiene relación con la cantidad de información disponible. Siguiendo con el ejemplo de la sección anterior, al evaluar la misma enzima O14172 pero esta vez en una base de datos de mayor volumen (TrEMBL), el resultado no es el mismo; la mejor alineación ya no corresponde a la enzima B8GUD3 (con la cual tiene un 36 % de identidad), sino con la enzima S9W3Q0 (con un 60.3 % de identidad). Existen secuencias que tienen mejores alineaciones en bases de datos más grandes (TrEMBL), lo que conlleva un problema puesto que puede que secuencias que BLAST clasifica de manera errónea usando SwissProt

sean bien clasificadas usando TrEMBL; por otro lado, si BLAST sigue sin poder asignar la clase correcta, puede que esta se esté comparando con nuevas clases que en SwissProt no existen. El problema no es el modelo en sí, sino la información de la que se dispone: ante una nueva secuencia, el algoritmo propuesto clasifica basándose en los datos de entrenamiento, forzando a extraer información de secuencias con las que el porcentaje de identidad al alinear puede que no sea alto, pero es lo más similar con que se cuenta. Si en un futuro se cura manualmente y se incorpora información de TrEMBL a SwissProt, la primera etapa del modelo que consta del uso de BLAST podría entregar distintos resultados al tener más y mejores posibilidades de realizar alineaciones, eliminando los casos en que la clase predicha difiere bastante de lo que se espera. A su vez, podría haber clases que hoy en día se clasifican con un 100 % por BLAST que tengan errores lo que daría pie a la posibilidad de mejorarlas. Para ello, bastaría con volver a entrenar el modelo.

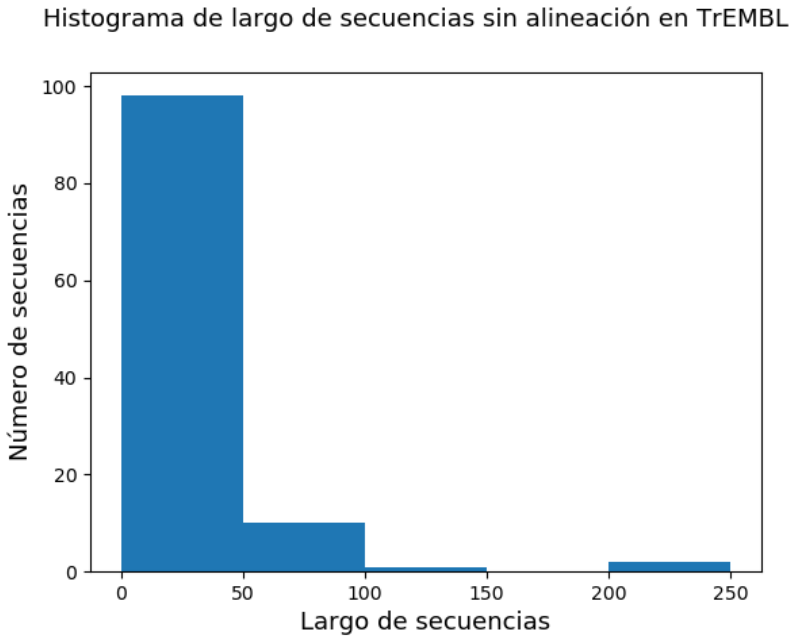


Figura 6.2: Histogramas de largos de secuencias no alineadas en TrEMBL.

Capítulo 7

Conclusiones y trabajo futuro

La mayoría de los trabajos relacionados a la anotación funcional de enzimas proponen un método alternativo a las tradicionales búsquedas por homología, sin considerarlos debido a sus conocidas limitaciones. Sin embargo se ha visto que dichos acercamientos como BLAST aún cuentan con un fuerte poder discriminatorio, puesto que la asignación de función por medio de homología tiene buenos resultados en muchos casos. El aporte presentado en el presente trabajo es una alternativa a lo visto comúnmente en la literatura: BLAST-KNN. En vez de dejar completamente de lado los algoritmos basados en homología, se busca mejorar los casos en que estos presentan dificultades. De esta manera se tiene un predictor que combina la fuerza de la asignación de funcionalidad por medio de homología con las ventajas que entrega el uso de diversas fuentes de información para suplir los casos en que la similitud entre secuencias es baja. Por otro lado, se valida a InterPro como una plataforma con información muy relevante al momento de clasificar enzimas, en especial algunas de las bases de datos a las que tiene acceso como PANTHER y CDD, así como la misma clasificación que propone la plataforma (descriptores IPR), los cuales si bien por si solo no son tan efectivos, si lo son al combinarse con los descriptores de las otras bases de datos. La combinación de toda esa información presenta una nueva manera de representar enzimas como un vector de largo fijo, puesto que según lo revisado en la literatura, o bien se utilizan solo los descriptores propuestos por InterPro o las distintas bases de datos por separado. En este trabajo se experimentó con cada uno de los casos, mostrando que el uso de ambas incrementa el rendimiento del modelo. Además, el algoritmo propuesto actúa como control de calidad en los casos de enzimas que pertenecen a clases con más de un número EC asociado, teniendo una alta tasa de clases mejoradas con respecto al uso de BLAST, dejando a criterio del experto los casos en que realmente existen clases que comparten algunos EC's pero difieren en otros, o si es que son el mismo conjunto y tan solo falta verificar si el ensamble de las secuencias está bien realizado y no le faltan fragmentos que puedan indicar una nueva funcionalidad, que se agregue a las ya conocidas.

Se valida a BLAST como un método efectivo para separar proteínas entre enzimas y las que no lo son. Por otro lado y bajo riesgo de sobreajuste, se clasifica al cuarto dígito EC en la gran mayoría de las clases existentes, incluyendo todas aquellas que contengan al menos 2 representantes, esto pues se puso énfasis en lograr una representación compacta y poderosa de las enzimas, además del uso de un seleccionador de características, lo que se comprueba con la gran cantidad de clases que son separables por medio de solo una característica. Este intento de clasificación global representa otra ventaja del modelo frente a otros trabajos que se reducen a pequeños conjuntos de datos, dejando la duda de su capacidad generalizadora.

Dentro de las limitaciones del algoritmo propuesto, la más fuerte es la información disponible, lo que se traduce en una exclusión de todas aquellas clases EC con un solo representante. Además, influye en el hecho de que las hipótesis realizadas en este trabajo puedan verse alteradas al utilizar una base de datos de mayor envergadura como TrEMBL. Esto se vio al realizar la prueba sobre datos fuera de SwissProt, en que en varios casos, la mejor alineación encontrada por BLAST difería de acuerdo a la base de datos utilizada, encontrando secuencias más similares en bases más grandes como TrEMBL. Es por ello que se sugiere volver a entrenar el modelo cada 6 meses, dejando esa ventana de tiempo para que SwissProt se actualice con información de TrEMBL. Por otro lado, existen casos en que el modelo no puede predecir dado que los descriptores extraídos de InterPro para algunas enzimas particulares no corresponden a los comúnmente visto en la clase a la que se asigna. Es por ello que junto a la actualización que se hace de los datos, se debe actualizar InterPro para obtener nuevos descriptores que puedan describir de mejor y más unificadamente a una clase. Se sugiere la misma ventana de tiempo que para SwissProt.

Como trabajo futuro y con la intención de mejorar la capacidad predictiva del modelo, se sugiere mejorar la manera de describir las enzimas. La representación de los datos es un paso esencial en el aprendizaje de máquinas. Con una buena representación, hasta los modelos más simples pueden lograr buenos resultados. Una alternativa es combinar la información de InterPro con las características que se pueden extraer de la secuencia, como las disponibles en ProFET. Para estas últimas, un nuevo acercamiento puede ser extraer características tanto de la secuencia completa como de ciertos fragmentos que representen algo importante, dado que se sabe que no todo el largo de la secuencia es igualmente informativa. De esta manera se podrían encontrar dominios importantes.
Lo anteriormente propuesto implica un aumento en el número de características usadas para describir una secuencia. Es por ello que se deben utilizar métodos más poderosos para seleccionar características. Se propone el uso de variantes de teoría de la información como *Joint Mutual Information Maximisation* [44] que se basan en la idea de seleccionar aquellas características de mayor relevancia, pero a la vez minimizando la redundancia entre ellas utilizando probabilidades conjuntas.

La clasificación EC es jerárquica. En este trabajo solo se utilizaron aquellas enzimas con todos sus números EC. Sin embargo, para incorporar aquellas que fueron dejadas fuera y poder predecir distintos dígitos, se propone la extensión a un clasificador *multilabel*, como es el caso de la variante *ML-KNN* que realiza un KNN por cada etiqueta, pudiendo entregar un ranking de ellas. También se podría utilizar algún algoritmo más poderoso como SVM *structured output*, sin embargo, habría que tener cuidado con los casos con muy pocas muestras. En ambos casos se deben cambiar las métricas de evaluación por algunas especiales para casos *multilabels* (por ejemplo, *HammingLoss* [45]).

Por último, aquellas secuencias no alineadas pueden utilizarse directamente en algún clasificador, con el fin de tener algún tipo de información sobre ellas. Para esto se debe incorporar un paso extra luego de usar BLAST, en que se use InterPro sobre estos casos para luego clasificar. Para no clasificar sobre todas las clases disponibles y ahorrar cálculo computacional, se pueden reducir a aquellas clases en que existe baja identidad entre los miembros de la misma. Esto se puede determinar usando CD-HIT, eligiendo un umbral de identidad. Se sugiere 40 % de identidad como umbral, dado lo propuesto en trabajos como [43].

Bibliografía

[1] Libbrecht MW. et al. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16:321–332, 2015.

[2] Raschka S. *Python Machine Learning*. Packt Publishing, 2015.

[3] Jiang Y. et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(184), 2016.

[4] Tetko IV. et al. Beyond the 'best' match: machine learning annotation of protein sequences by integration of different sources of information. *Bioinformatics*, 24(5):621–628, 2008.

[5] The UniProt Consortium. Uniprot: the universal protein knowledgebase. 45:158–169, 2017.

[6] Pearson W.R. *An Introduction to Sequence Similarity ("Homology") Searching*. Current Protocols in Bioinformatics, 2013.

[7] Altschul S.F. Gapped blast and psi-blast: A new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

[8] McDonald AG. and Tipton KF. Fifty-five years of enzyme classifications:advances and difficulties. *FEBS J.*, 281:583–592, 2014.

[9] Johnson LS. et al. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC Bioinformatics*, 11:431, 2010.

[10] Cai CZ. et al. Svm-prot: Web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucleic Acids Research*, 31:3692–3697, 2003.

[11] Ferrari L. et al. Enzml: multi-label prediction of enzyme classes using interpro signatures. *BMC Bioinformatics*, 13(61), 2012.

[12] Lan L. et al. Ms-knn: protein function prediction by integrating multiple data sources. *BMC Bioinformatics*, 14(Suppl 3), 2013.

[13] Asgari E. et al. Continuous distributed representation of biological sequences for deep proteomics and genomics. 10(11), 2015.

[14] Li YH. et al. Svm-prot 2016: A web-server for machine learning prediction of protein functional

families from sequence irrespective of similarity. *PLoS ONE*, 11(8), 2016.

[15] Nagao C. et al. Prediction of detailed enzyme functions and identification of specificity determining residues by random forests. *PLos ONE*, 9(1), 2014.

[16] Min S. et al. Deep learning in bioinformatics. *Briefings in Bioinformatics*, pages 1–19, 2016.

[17] Webb E.C. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes*. Academic Press, 1994.

[18] Henikoff S. & Henikoff J.G. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA*, 89(22):10915–10919, 1992.

[19] Kadam K. et al. Prediction of protein function based on machine learning methods: An overview. 2014.

[20] Tobar F. Máquinas que aprenden. *Bits de ciencia*, 15:36–43, 2017.

[21] Tang J. et al. *Feature Selection for Classification: A Review*, chapter 2, pages 37–64. Data Classification Algorithms and Applications, 2014.

[22] Kullback S. et al. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[23] Silverman B.W. and Jones M.C. E. fix and j.l. hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951). *International Statistical Review*, 57:233–238, 1989.

[24] Cortes C. et al. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[25] Ho T.K. Random decision forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pages 278–282, 1995.

[26] Arlot S. et al. A survey of cross-validation procedures for model selection. *Statist. Surv.*, 4:40–79, 2010.

[27] Cawley G. et al. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

[28] Wang T. et al. Domsign: a top-down annotation pipeline to enlarge enzyme space in the protein universe. *BMC Bioinformatics*, 16(96), 2015.

[29] Wang Y. et al. Support vector machine prediction of enzyme function with conjoint triad feature and hierarchical context. *BMC System Biology*, 5, 2011.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[31] Szalkai B. & Grolmusz V. Near perfect protein multilabel classification with deep neural networks. *Methods*, 2017.

[32] Cock PJ. et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422–1423, 2009.

[33] Jones E. et al. Scipy: Open source scientific tools for python. 2001.

[34] Pedregosa F. et al. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, vol 12, pages 2825-2830 edition, 2011.

[35] Ofer D. & Linial M. Profet: Feature engineering captures high-level protein functions. *Bioinformatics*, 31(21):3429 – 3436, 2015.

[36] Varshavsky R. et al. When less is more: improving classification of protein families with a minimal set of global. In *Algorithms in Bioinformatics: 7th International Workshop*, pages 12–24, 2007.

[37] Dubchak I. et al . Prediction of protein folding class using global description of amino acid sequence. *Proc.Natl.Acad.Sci.*, 92(19):8700–8704, 1995.

[38] Fu L. et al. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[39] Buchfink B. et al. Fast and sensitive protein alignment using diamond. *Nature Methods*, 12:59–60, 2015.

[40] Finn D. et al. Interpro in 2017 — beyond protein family and domain annotations. *Nucleic Acids Research*, 45:190–199, 2017.

[41] Jones P. et al. Interproscan 5: genome-scale protein function classification. *Bioinformatics*, 30(9):1236–1240, 2014.

[42] Schläpfer P. et al. Genome-wide prediction of metabolic enzymes,pathways, and gene clusters in plants. *Plant Physiol.*, 173(4):2041–2059, 2017.

[43] Skolnick J. et al. How well is enzyme function conserved as a function of pairwise sequence identity? *J Mol Biol.*, 333:863–882, 2003.

[44] Bennasar M. et al. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42:8520–8532, 2015.

[45] Diéz J. et al. Optimizing different loss functions in multilabel classifications. *Progress in Artificial Intelligence*, 3:107–118, 2015.

[46] Akhtar MN. et al. Evaluation of database search programs for accurate detection of neuropeptides in tandem mass spectrometry experiments. *J Proteome Res*, 11:6044–6055, 2012.

[47] Kelley LA. et al. The phyre2 web portal for protein modeling, prediction and analysis. *Nature Protocols*, 10:845–858, 2015.

[48] Grosdidier A. et al. Swissdock, a protein-small molecule docking web service based on eadock dss. *Nucleid Acid Res.*, 39:207–277, 2011.