DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

LigQ 2.0 y ReverseLigQ: dos herramientas de Virtual Screening e Inverse Virtual Screening basadas en aprendizaje automático

Tesis de Licenciatura

Ángel Abregú
aj.abregu@gmail.com

Martín de Micheli
shmdm7@gmail.com

Director
Pablo Turjanski
pturjanski@dc.uba.ar

Co-Director
Marcelo Martí
marcelo@qi.fcen.uba.ar

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Dedicado...

A nuestrxs colegas, que en su mayoría han sido compañerxs de cursada, por todo el apoyo moral recibido durante el desarrollo de este trabajo.

Angel:
A mi mamá y mi papá por el cariño, el esfuerzo y por estar presentes desde el principio, muchas gracias. A mis hermanas, mi abuela, madrina, padrino, prima, primos, por su cariño y apoyo incondicional.
A Rioly Valenzuela Olivera, por acompañarme en los momentos difíciles, en una etapa de crecimiento académico y personal, incentivándome a superarme a mi mismo cada día. A Alexis García Ayerdi y Saúl Quispe, por la compañía durante todos estos años de carrera.
A mi profesor de secundaria Gerardo Mamani, por haberme introducido al mundo de la programación y presentarme la carrera de Ciencias de la Computación. Muchas gracias.
A mi compañero de tesis Martín De Micheli, por su perseverancia y proactividad.

Martín:
A mi familia y amigos de toda la vida, por estar siempre. A mis padres, que supieron cultivar la curiosidad en mí.
A mis compañéros de cursada Rogelio Mita, Federico Pousa, Leopoldo Taravilse, Karina Borgna, Felipe Schargorodsky, Agustina Ciraco, Facundo Carrillo, Andrés Gutter, Mariano Cerruti, Rodrigo Castaño, Julian Peller, Adrián Tamburri, Dardo Marasca, Brian Curcio, Pablo Antonio, Pablo Herrero y muchos otros que me aportaron muchísimo en mi formación académica y personal.
A todos los integrantes de la UTI, quienes estuvieron y quienes ya migraron a otros rumbos, por haberme formado profesionalmente.
A mi compañero de tesis Ángel Abregú, por todos los días de esfuerzo y creatividad para llevar a cabo este trabajo.

Agradecimientos

A nuestros directores. A Pablo Turjanski por confiar en nosotros y por su dedicación extrema hasta el último minuto. A Marcelo Martí, Germán Burguener y la Plataforma Bioinformática Argentina (BIA) por guiarnos en el desarrollo de la tesis dentro de un área de la que poco conocíamos y por poner a nuestro alcance los recursos necesarios que hicieron posible este trabajo. Muchas gracias.

A Leandro Radusky por darnos una mano desde el otro lado del Atlántico para dilucidar el funcionamiento interno de LigQ y varios conceptos de bioinformática.

A los jurados Esteban Lanzarotti y Darío Fernández Do Porto, por las correcciones y las recomendaciones para mejorar la calidad de nuestro trabajo.

A Ariel Berenstein por brindarnos una visión práctica y el uso apropiado de herramientas y técnicas de análisis estadístico.

A lxs docentes de la carrera de Ciencias de la Computación. A Carolina Curátolo, por sus talleres de inglés que nos fueron de mucha ayuda a la hora de redactar y preparar una presentación.

A la Unidad de Tecnologías de la Información (UTI) por la paciencia hasta recibirnos y por permitirnos hacer uso de los equipos para ejecutar muchos de los programas cuando hizo falta.

Al Centro de Cómputos de Alto Rendimiento (CeCAR) por concedernos el uso de recursos computacionales que nos permitió realizar parte de los experimentos incluidos en este trabajo.

Al DC, a Exactas, a la UBA y a una educación pública, gratuita, y de calidad.

¡Gracias!

Resumen

En el presente trabajo de tesis se aborda el problema de generación de conjuntos de moléculas pequeñas que puedan comportarse como ligandos para proteína de interés y su inverso, la generación de conjuntos de proteínas con altas chances de acoplamiento a un ligando objetivo. En la actualidad, las bases de datos de moléculas poseen millones de compuestos y, poder predecir acoplamientos, es un problema relevante para la comunidad. Durante el desarrollo de esta tesis, nos enfocamos específicamente en el problema de vincular las propiedades conocidas sobre el sitio de unión de la proteína estudiada con las propiedades de los ligandos candidatos. Los descriptores escogidos son, por lo general, valores fisicoquímicos, morfológicos y topológicos, ya sean calculados o conocidos experimentalmente. Para realizar este vínculo de manera óptima, aplicamos técnicas de aprendizaje automático, estableciendo una valoración de los compuestos obtenidos en base a los parámetros calculados como óptimos para que una molécula se acople eficientemente a nuestro objetivo de interés. Los resultados finales, si bien no fueron los esperados, dan lugar a modelos de uso real mediante su incoporación en sistemas web que permiten su uso irrestricto.
Para acercar a la comunidad nuestros desarrollos, tomamos como punto de partida una herramienta preexistente denominada LigQ. Esta herramienta permite calcular, dada una proteína como entrada, conjuntos de moléculas candidatas a ser ligandos de la misma, es decir, que pueden potencialmente acoplarse a ella modulando o inhibiendo su actividad. Adaptamos dicha herramienta, incorporando modelos de aprendizaje automático construídos durante este trabajo. Análogamente, para el problema inverso, desarrollamos una herramienta similar, a la que denominamos reverse-LigQ.

Palabras clave Bioinformática, Quimioinformática, Biología Computacional, Redes Neuronales, Aprendizaje Automático, Virtual Screening, Inverse Virtual Screening

Abstract

The present thesis work addresses the problem of generating sets of small molecules that can function as ligands for a certain protein of interest and its inverse, the generation of sets of proteins with high chances of binding to a target ligand. Currently, molecule databases have millions of compounds and it is a major problem for the scientific community to be able to predict these kind of bindings. During the development of this thesis, we focused specifically on the problem of establishing a relationship between the known properties on the binding site of the studied protein and the properties of the candidate ligands. The chosen descriptors are, in general, physicochemical, morphological and topological values, whether computationally calculated or experimentally determined. We aimed to infer the relationship between those properties by applying Machine Learning techniques, selecting the obtained compounds based on the parameters calculated as optimal for a molecule to efficiently bind our target of interest. The final results, although not as good as expected, gave birth to models of real use through its incorporation into web systems that allow its unrestricted use.

To bring our developments closer to the scientific community, we made use of a pre-existing tool called LigQ, which allows us to calculate, given a target protein as input, a set of molecules that can potentially bind to it and may ultimately modulate or inhibite its activity. We modified this tool by incorporating to it the Machine Learning models built in this work. Similarly, we developed a separate tool to address the inverse problem, which we called reverse-LigQ.

Keywords Bioinformatics, Chemoinformatics, Computational Biology, Neural Networks, Machine Learning, Virtual Screening, Inverse Virtual Screening

Contents

1 Introducción

El desarrollo de herramientas computacionales para el cálculo y análisis de datos se encuentra actualmente en constante expansión en diferentes campos de la ciencia. Particularmente en el campo de las ciencias de la vida, dada la cantidad de datos disponibles (generados por nuevos métodos experimentales) surge como una necesidad fundamental la implementación de técnicas computacionales para el análisis y manejo de datos, y su transformación en conocimiento.
Dentro del campo de la bioinformática estructural de proteínas y la quimioinformática, nos hemos concentrado en la generación, extensión y aplicación de herramientas en problemas relacionados con la salud humana.

1.1 El campo de la bioinformática

La bioinformática como concepto tiene límites difusos, no existe una única definición para su alcance y significado[1]. Consideraremos el área de la bioinformática, en un sentido amplio, como el conjunto de técnicas informáticas utilizadas para el almacenamiento, análisis y procesamiento de datos biológicos. Los primeros usos de estas técnicas se remontan al procesamiento de secuencias de ADN, el soporte físico de la información genética. Ya a finales de los años setenta del siglo XX, el volumen de datos generados por secuenciadores volvió rápidamente poco práctico cualquier método manual para el procesamiento de su información[2], lo cual llevó a que se convirtiese en una de las áreas más prolíficas en cuanto al desarrollo de técnicas informáticas para usos aplicados a la biología. El alineamiento de secuencias (método muy extendido para compararlas) ha permitido inferir relaciones evolutivas[3], homología entre proteínas[4], asignar funciones a nuevas proteínas[5], inferir su estructura tridimensional[6,7] y el plegado de proteínas[8,9], entre otros resultados relevantes.

Otra aplicación extendida de la bioinformática es el descubrimiento y diseño de fármacos, y se basa en el establecimiento de relaciones entre las propiedades pertenecientes a moléculas orgánicas pequeñas (fármacos) y aquellas macromoléculas con las cuales potencialmente se pueden acoplar, modulando o inhibiendo su actividad dentro de la célula. Por lo general, los blancos de los fármacos son las proteínas, las cuales son clave en procesos metabólicos asociados a una enfermedad o patología específica, o a la infectividad o supervivencia de algún microbio patogénico. Si bien no necesariamente son causantes de la enfermedad, sí tienen la capacidad de modificarla de alguna manera[10]. El rol que cumple la bioinformática en estos estudios es principalmente el de contribuir a identificar o diseñar pequeñas moléculas (ligandos) que tengan la capacidad de enlazarse con estos targets o blancos y así modular o inhibir su función biológica[11]. Una de las metas de nuestro trabajo será desarrollar técnicas que permitan predecir la afinidad entre los blancos proteicos y los ligandos, a partir de analizar las distintas características de cada uno de ellos.

1.2 Proteínas y sus funciones

Las proteínas son macromoléculas. Estructuras compuestas de unidades llamadas aminoácidos. Son, por lejos, las biomoléculas más versátiles y diversas; y de particular interés para la biología por el papel fundamental que desempeñan para la vida. Suponen aproximadamente la mitad del peso de los tejidos del organismo[12], están presentes en todas las células del cuerpo humano y participan en prácticamente todos los procesos biológicos. Un modesto catálogo comprende a las *hormonas*, reguladoras de la actividad celular; la *hemoglobina*, con funciones de transporte de oxígeno en sangre; los *anticuerpos*, encargados de acciones de defensa natural contra infecciones o agentes patógenos; la mayor parte de las *enzimas*, catalizadoras de reacciones químicas en organismos vivientes; y los *receptores* de las células, a los cuales se acoplan moléculas externas capaces de desencadenar una respuesta determinada[13].
No es de extrañar, por lo tanto, que la posibilidad de regular estas funciones constituya una de las quimeras de la biología moderna. En esta dirección, si la noción de función biológica está asociada a la noción de interacción con otras moléculas, estudiar las capacidades de las proteínas para establecer uniones moleculares constituye el quid de la cuestión.
El conocimiento actual establece que, dentro de un sinnúmero de características de las proteínas, es sin duda su estructura tridimensional lo que les permite la versatilidad para participar en una amplia gama de interacciones químicas[14].
Los aminoácidos de una proteína se ordenan en diversas formas dando lugar a su *estructura tridimensional*, i.e. la disposición en el espacio de los átomos que la componen. Suelen diferenciarse en este caso cuatro niveles de estructuración[15], descriptos de manera gráfica en la Figure 1.1:

- La **estructura primaria** de una proteína, definida por la secuencia de aminoácidos de la cadena polipeptídica que la compone (o cadenas, si está formada por más de un polipéptido).

- La **estructura secundaria**, la disposición espacial local que ocupa "la columna vertebral" (*backbone* en inglés, es decir la porción de la molécula compartida por cada uno de los aminoácidos) de la proteína, y se determina mediante la conformación de enlaces tipo puente de hidrógeno. Existen estructuras secundarias bien definidas y ordenadas, como hélices α, hojas plegadas β y regiones desestructuradas.

- La **estructura terciaria** se define como la estructura que adopta la cadena polipeptídica en el espacio. El modo en que la secuencia de aminoácidos se pliega en el espacio (de forma globular, como fibra, etc) determina el o los dominios de plegado que pueden ser asignados a la proteína. Se denomina dominio estructural a un elemento constitutivo (o unidad) de la estructura de las proteínas que estabiliza su plegado de manera independiente. Las proteínas poseen, por lo general, uno o más dominios de plegado.

- Por último, la **estructura cuaternaria** se forma al unirse varias cadenas polipeptídicas con estructura terciaria, mediante enlaces débiles. De este modo conforman lo que se conoce como "complejo proteico".

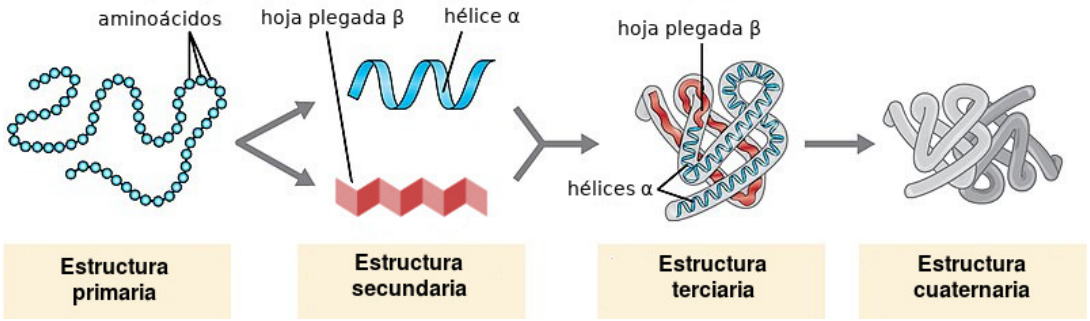


Figure 1.1: Cuatro niveles de organización de la estructura proteica. *(Fuente: adaptado de Biology LibreTexts)*[16].

Cavidades

Dentro de la estructura proteica pueden existir, por un lado, regiones que se pliegan sobre sí mismas (dominios con estructuras globulares), por lo tanto poco accesibles físicamente; y por otro, regiones que no presentan plegamientos y permanecen accesibles a otros agentes[17]. Es en estas últimas donde se forman cavidades o "bolsillos" (en inglés, *pockets*), las cuales dependiendo de sus propiedades físico-químicas (volumen, carga, etc.) pueden dar lugar a acoplamientos con moléculas pequeñas que denominaremos ligandos[18,19]. Estos acoplamientos son determinados por residuos de aminoácidos próximos en el espacio tras el plegado, lo cual no es posible de determinar si miramos únicamente la estructura primaria de una proteína, es decir, su secuencia, en donde pueden estar muy distantes[20]. De esto surge la importancia de conocer la estructura 3D de una proteína, dado que generalmente permite obtener más información sobre su función que la secuencia misma [a].

1.3 Ligandos y sus propiedades

Los ligandos son moléculas con capacidad de unirse químicamente a, por ejemplo, otras (bio)moléculas como las proteínas, produciendo cierto impacto en la bioquímica celular. Se acoplan a una cavidad o bolsillo de éstas y, generalmente, alteran su conformación química modificando su estructura tridimensional y, por ende, su estado funcional[21]. De este modo, el ligando termina actuando como catalizador, inhibidor, activador o neurotransmisor[22]. Las drogas son un caso particular de ligandos exógenos, es decir, provenientes del exterior del organismo[23].
Es importante aclarar que, debido a que el enlace con la proteína receptora involucra fuerzas intermoleculares débiles (como por ejemplo enlaces iónicos, puentes de hidrógeno y fuerzas de Van der Waals), éste no cambia sustancialmente la composición química de la última. Inclusive, este acoplamiento (en inglés, *binding*) es en algunos casos reversible, mediante un proceso conocido como *disociación.*[22] Para medir entonces cuán fuerte es este enlace se utiliza el concepto de afinidad, el cual tipifica la tendencia o fuerza de estos cambios funcionales, y no está dado sólo por las características del receptor y el ligando, sino también por factores externos como los solventes que interactúan en la reacción[24]. Éstos últimos proveen un medio químico que regula la capacidad de la proteína y el ligando para enlazarse; lo cual cobra mucha relevancia

[a] Los métodos más utilizados para obtener la estructura tridimensional de una proteína se encuentran detallados en el subsection A.1.1

considerando que los estudios en el área se focalizan casi exclusivamente en procesos que ocurren dentro del organismo, donde las condiciones pueden ser muy variables, así como entre individuos de una misma población[23]. De esta manera, se hace imprescindible el estudio de los procesos generales que dominan el funcionamiento del organismo como un todo. El área que comprende estos fenómenos se denomina *farmacocinética*, y se define como "el estudio de los procesos a los que un fármaco es sometido a través de su paso por el organismo, desde el momento en el que es administrado hasta su total eliminación del cuerpo"[25], que depende principalmente de 4 factores que se abrevian habitualmente como *ADME*: Absorción, Distribución, Metabolismo y Excreción[26]. El proceso completo desde el descubrimiento de una droga hasta su liberación al mercado puede visualizarse esquemáticamente en la figura 1.2.

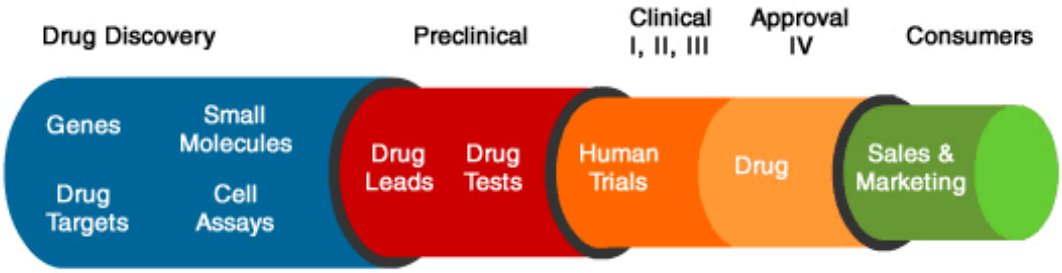


Figure 1.2: Proceso de desarrollo de una nueva droga. En primera instancia (azul) se selecciona cuáles serán los blancos, un conjunto grande de moléculas candidatas, y sobre qué líneas celulares se harán ensayos. A continuación (en rojo), compuestos que tienen buena probabilidad de tener afinidad con el blanco proteico son probados de manera experimental. Finalmente, se realizan experimentos *in vivo* (amarillo) antes de llegar la etapa de comercialización del compuesto (verde). Imagen extraída de *Bioinformatics and drug discovery*: `http://crdd.imtech.res.in/indipedia/index.php/Bioinformatics_and_Drug_discovery`

1.4 *Virtual Screening*

A lo largo del tiempo, las drogas se han desarrollado identificando los ingredientes activos de sustancias tradicionalmente ya utilizadas, o mediante descubrimientos principalmente azarosos (serendipia). El análisis de células u organismos completos permitió identificar sustancias con efecto terapéutico, en un proceso conocido como **farmacología clásica**[27]. Sin embargo, la falta de modelos que representen con exactitud las condiciones apropiadas o las respuestas esperadas de una forma reproducible y económica conllevan a que el proceso de descubrimiento y desarrollo de nuevos fármacos sea un proceso caro, difícil e ineficiente. Acertar la eficiencia y toxicidad que tendrá una molécula representa una porción considerable del proceso de desarrollo, muchas veces logrado mediante el uso de modelos no humanos[28]. El proceso completo puede durar más de una década, y la tasa de éxito es baja. Es por eso que es de suma importancia para los laboratorios actuales desarrollar técnicas que permitan pre-seleccionar moléculas para actuar como fármacos en un caso determinado. Tal es el caso del **cribado virtual** (en inglés, *Virtual Screening*[b]), un filtrado computacional de moléculas para su posterior evaluación experimental. No obstante deba validarse con ensayos experimentales *in silico*, la técnica reduce significativamente el número de ensayos que se harían si no hubiera una selección previa de compuestos.

[b] Si bien las ideas detrás del nombre *Virtual Screening* ya contaban con presencia en la comunidad científica durante los años '90s, el término fue acuñado en el año 1998 en el trabajo *Virtual screening—an overview*[29].

Existen diversos filtros que se utilizan para llevar a cabo el cribado virtual, los cuales pueden variar según la complejidad de la base de datos y la información experimental de la que se disponga. Por ejemplo, si se conoce la estructura tridimensional del receptor se sugiere un cribado basado en la estructura (acoplamiento molecular). Si solo se conocen los compuestos activos, pero no el receptor, entonces la búsqueda se hace basada en el ligando (similitud molecular). Si se conoce la estructura 3D del receptor y de los compuestos activos se pueden combinar filtros para facilitar la búsqueda, tales como descriptores moleculares y propiedades farmacocinéticas, entre otros. En suma, las técnicas para hacer cribado virtual dependen de la información disponible [30].

1.5 *Inverse Virtual Screening*

Conocer a qué proteínas, además de la objetivo, es capaz de acoplarse un ligando es crucial para evitar efectos adversos en el organismo. No sólo puede ayudar a reducir la tasa de fracaso de un proyecto dentro del proceso de desarrollo de drogas, sino que también puede brindar la oportunidad de dar un uso más amplio a moléculas ya estudiadas (ya sea como drogas multiobjetivo, o revelando mecanismos de acción desconocidos) [31]. Los avances en el secuenciamiento genómico y el campo de la bioinformática permitieron comenzar a operar en el camino inverso al de *virtual screening*: dado un compuesto, predecir posibles targets biológicos a los cuales pueda acoplarse. Se ha vuelto una práctica habitual analizar grandes conjuntos de datos de compuestos químicos probados contra objetivos biológicos aislados, identificados (al menos hipotéticamente) con algún rol en una enfermedad [32]. La aplicación de la bioinformática en esta labor (denominada **farmacología inversa**) es esencial para lograr manejar la cantidad de información necesaria y brindar soluciones que permitan acelerar y abaratar el desarrollo de nuevos fármacos [27,33,34].

1.6 Sistemas de recomendación

Para poder brindar predicciones como las requeridas por Virtual Screening o Inverse Virtual Screening, es necesario contar con técnicas que analicen los datos existentes y permitan establecer ciertas relaciones entre propiedades de proteínas y ligandos. Existe un conjunto de herramientas que sirven a este propósito conocidas como sistemas de recomendación, un conjunto de técnicas y programas de software que brindan sugerencias de ítems con mayor probabilidad de resultar de interés a un usuario [35].

Uno de los primeros sistemas de recomendación, Tapestry [36], fue desarrollado en Xerox Palo Alto Research Center y publicado en Communications en el año 1992. La motivación que llevó a la creación de Tapestry (que significa *tapiz* o *entramado* en inglés) estaba asociada al uso creciente del correo electrónico, lo que resultaba en que los usuarios "se vean inundados de un flujo enorme de documentos". El sistema pretendía aminorar el trabajo de los usuarios utilizando lo que hoy conocemos como filtro colaborativo, personas colaborando para ayudar a personas a realizar el filtrado. El sistema únicamente requería que los usuarios grabaran sus reacciones a los documentos que leían. El nombre que el sistema formalmente le daba a estas reacciones era "anotaciones" (algo que hoy en día conocemos como etiquetado o *tagging*). Por ejemplo "ese artículo me hizo reír, así que lo etiqueto como 'gracioso' ". Un filtro típico para un usuario, digamos Raúl, redundaba en "cada vez que Ana, Carlos o Alberto etiquetan algo como 'humor sutil' lo reenvío a la casilla de Raúl", presumiblemente porque Raúl solía disfrutar de dichos artículos y etiquetarlos a su vez como 'humor sutil'. El sistema tuvo relativo éxito pero

principalmente dio la pauta de que podía utilizarse el historial de datos para resolver esta clase de problemas. A partir de esto se comenzaron a desarrollar diversas herramientas para realizar este tipo de análisis, siendo las más conocidas actualmente las denominadas de **aprendizaje automático** (en inglés, Machine Learning). Como su nombre lo indica, estas técnicas, buscan crear modelos computacionales que "aprendan" de la experiencia de casos existentes para poder inferir qué sucederá cuando un nuevo caso se presente. Utilizando esta idea propondremos implementar un software que dada una proteína objetivo nos sugiera ciertos compuestos como ligandos candidatos y, por otro el problema inverso, sugiriendo proteínas en función de un compuesto objetivo.

1.7 LigQ

Existe más de una herramienta de Virtual Screening. En particular, nos vamos a centrar en LigQ[37], como se menciona en su página web[c]: *"A Webserver to Select and Prepare Ligands for Virtual Screening"* (un servidor web para seleccionar y preparar ligandos para cribado virtual). La razón principal de esta elección viene dada por el hecho de que esta herramienta fue desarrollada en el marco de una tesis doctoral en colaboración con la **Plataforma Bioinformática Argentina**[38] (abreviada BIA), de forma que contamos con acceso pleno al código fuente y contacto directo con su desarrollador.
La arquitectura de LigQ comprende una serie de módulos que se conectan en cadena, formando un pipeline. La ejecución toma como entrada un identificador de una proteína y devuelve como resultado un conjunto de compuestos candidatos. Si bien se explicará con mayor detalle todo lo relacionado a LigQ en la chapter 4, lo interesante de destacar en primera instancia es que actualmente LigQ utiliza conceptos como familias de proteínas y métricas de similitud entre ligandos para realizar sus recomendaciones, pero no modelos de aprendizaje automático como los anteriormente mencionados. Incorporar este tipo de técnicas a LigQ es uno de los objetivos de este trabajo de tesis.

1.8 Organización de este trabajo de tesis

Los objetivos de este trabajo pueden entonces resumirse en los siguientes ítems:

- estudiar los datos públicos de proteínas y compuestos químicos y proveer un dataset de calidad, que nos permita construir buenos modelos de recomendación proteína-ligando
- aplicar técnicas de aprendizaje automáticos sobre este conjunto de datos para construir modelos de predicción de interacción proteína-ligando
- integrar los modelos de predicción a herramientas de Virtual Screening e Inverse Virtual Screening

En el chapter 2 se describirá todo el proceso de obtención y refinamiento de los datos que luego utilizaremos para entrenar a nuestros modelos de aprendizaje automático, detallados en los chapter 3 y chapter 5. En el chapter 4 se explicará cómo se incorporó el esquema de predicción a la herramienta LigQ, junto con las modificaciones que fueron hechas sobre la arquitectura original

[c] http://ligq.qb.fcen.uba.ar/

y las funcionalidades del producto final. En último lugar, en el chapter 6 se presentará la implementación de una herramienta de Inverse Virtual Screening completamente nueva, desarrollada ad-hoc para este trabajo de tesis: **reverse-LigQ**.

2 | Confección del conjunto de datos

> "The thing is, all datasets are flawed. That's why data preparation is such an important step in the machine learning process. In a nutshell, data preparation is a set of procedures that helps make your dataset more suitable for use. And these procedures consume most of the time spent on machine learning. Sometimes it takes months before the first algorithm is built!" [39]

En el presente trabajo nos basamos en información actualmente existente de compuestos, proteínas y de las uniones conocidas entre ambos, para construir modelos que permitan predecir nuevos acoplamientos. Nos propusimos dos objetivos principales: por un lado, predecir compuestos con altas chances de acoplarse con una proteína dada; y por otro el caso inverso, predecir proteínas con buenas posibilidades de acoplarse con un compuesto dado. Para encarar ambos problemas decidimos emplear herramientas de aprendizaje automático, las cuales utilizan grandes conjuntos de datos (la literatura del área se refiere a estos conjuntos con el término *datasets*) con el fin de realizar cálculos estadísticos que permitan identificar patrones, estimar valores y/o realizar predicciones.

Como punto de partida, contamos inicialmente con un dataset provisto por el BIA, elaborado especialmente para nuestro caso de estudio. El proceso de curado de éste (a cargo también del grupo BIA y, en algunos casos, motivado por observaciones nuestras) fue derivando en distintas "versiones" de los datos, en las cuales se fueron aplicando múltiples filtros, los cuales se detallarán en las secciones subsiguientes. A su vez, estas versiones han sido numeradas, y si bien se hacen las distinciones correspondientes durante el procesamiento (*scripts*), en lo referente a la obtención de resultados finales descriptos en este informe, nos basamos únicamente en la última versión.

A continuación detallaremos las características de este conjunto inicial de datos. Detallaremos los diferentes análisis estadísticos realizados y las múltiples modificaciones de los datos que surgieron a partir de dichos análisis. Hacia el final del capítulo describiremos los archivos finales que fueron usados en la construcción de modelos computacionales que nos permitieron implementar herramientas de *Virtual Screening* de uso real y productivo.

2.1 Descripción del dataset incial

El dataset inicial está compuesto de registros provenientes de una base de datos biológica pública, el *Protein Data Bank* (abreviado `PDB`)[a]. Cuenta con una serie de descriptores o propiedades (en inglés, *features*) de proteínas y ligandos distribuídos en tres archivos csv:

- `ligands.csv`: características de compuestos químicos identificados como ligandos.
- `pockets.csv`: características de los pockets detectados en cada proteína.
- `bindings.csv`: acoplamientos entre pockets y ligandos presentes en los otros dos archivos.

[a] Para ver el detalle de las bases de datos y en qué formatos son guardados los registros, consultar el Apéndice A, sección A.2.1

Si bien hubo cambios en el nombre de estos archivos, la forma de estructurar los datos bajo tres conceptos (pocket, ligando y acoplamiento) fue la que decidimos mantener a lo largo de todo nuestro trabajo.

Propiedades de un ligando

La totalidad de los ligandos presentes en este dataset inicial provienen de PDB. Los descriptores elegidos por el BIA son los que existen por defecto en LigQ, que a su vez son los que se detallan para cada compuesto en la base de datos ZINC. A este conjunto se le agrega el número de anillos aromáticos, calculado a partir del SMILES del compuesto, que figura en PDB. El listado final puede verse en la tabla 2.1.

Nombre de la propiedad	**Nombre en la tabla**	**Descripción**
Molecular Weight	`mwt`	Peso molecular.
LogP	`log_p`	Logaritmo del coeficiente de partición.
LogS	`log_s`	Logaritmo de la solubilidad en agua.
DesolvPolar	`desolv_polar`	Energía de solvatación en solvente polar.
DesolvApolar	`desolv_apolar`	Energía de solvatación en solvente no polar.
Hydrogen Bond Donors	`hbd`	Cantidad de donantes de puentes de hidrógeno.
Hydrogen Bond Acceptors	`hba`	Cantidad de aceptores de puentes de hidrógeno.
Topological Polar Surface Area	`tpsa`	Suma de las superficies de átomos polares.
Charge	`charge`	Carga neta.
Number of Rotatable Bonds	`nrb`	Cantidad de enlaces que permiten la libre rotación alrededor de sí mismos.
Aromatic Rings	`arom_rings`	Cantidad de anillos aromáticos.

Table 2.1: Descriptores de ligandos presentes en el archivo `ligands.csv`. Una versión más detallada del significado de cada una de ellos figura en el Apéndice A, sección A.2.5.1.

Propiedades de una proteína

Para el caso de una proteína, existe una gran cantidad de descriptores disponibles en las bases de datos consultadas. Basta ver cualquier entrada de PDB para encontrar información que va desde los átomos que componen su estructura hasta el mapa de electrones de la molécula. Sin embargo, como fue mencionado anteriormente, el enfoque de este trabajo de tesis estuvo guiado bajo la hipótesis de que era más conveniente prestar atención a los bolsillos y sus descriptores, dado que es allí donde ocurren los acoplamientos. La decisión de qué características de éstos elegir estuvo influenciado por LigQ y la herramienta de detección de bolsillos que éste usa: **fpocket**[40][b]. El listado final de las propiedades que componen el archivo `pockets.csv` puede verse en la Table 2.2.

[b] Los detalles del algoritmo que **fpocket** usa para generar los pockets puede encontrarse detallado en el Apéndice A, sección A.2.7.

Nombre de la propiedad	Nombre en la tabla	Descripción
Score	`score`	Puntaje que `fpocket` asigna al bolsillo.
Druggability Score	`druggability`	Probabilidad de unirse a una molécula pequeña tipo droga.
Number of alpha spheres	`number_spheres`	Número de α-esferas.
Volume	`volume`	Volumen del pocket aproximado con el de las α-esferas.
Mean alpha-sphere radius	`mean_alpha_sphere_radius`	Radio promedio de las esferas usadas.
Mean alp. sph. solvent access	`mean_alpha_sphere_solvent_access`	Medida de qué tan "incrustada" están las α-esferas en la superficie.
Flexibility	`flexibility`	Factor B promedio.
Hydrophobicity Score	`hydrophobicity_score`	Medida de la hidrofobicidad del pocket.
Polarity Score	`polarity_score`	Medida de la hidrofilicidad del pocket.
Charge Score	`charge_score`	Medida de la carga del pocket.
Volume Score	`volume_score`	Medida de volumen del pocket.
Mean Local hydrophobic density Score	`mean_local_hydrophobic_density`	Medida de hidrofobicidad.
Alpha sphere density	`alpha_sphere_density`	Medida de la compacidad del pocket.
Maximum distance between center of mass and alpha sphere	`com_alphasphere_distance`	Distancia máxima entre centro de masa del pocket y una α-esfera.
Apolar alpha sphere proportion	`apolar_alpha_sphere_proportion`	Proporción de α-esferas no polares de un pocket.
Proportion of polar atoms	`proportion_polar_atoms`	Proporción de átomos polares.
Apolar SASA	`sasa_apolar`	Área de la superficie no polar del pocket.
Polar SASA	`sasa_polar`	Área de la superficie polar del pocket.
Total SASA	`sasa_total`	Área total de la superficie del pocket.

Table 2.2: Descriptores de bolsillos o *pockets* de una proteína, presentes en el archivo `pockets.csv`. Para ver una versión más detallada del significado de cada una de ellos consultar el Apéndice A, sección A.2.7.

2.1.1 Reducción de sobre-representación

Se habla de un *sesgo experimental* (en inglés, *experimenter bias* o *research bias*) cuando los investigadores seleccionan (con o sin intención) sujetos que tienen más probabilidades de generar resultados deseados. Por ejemplo, tomar poblaciones rurales para el estudio de la contaminación del aire a lo largo de nuestro país constituirá con serias dudas un grupo plenamente representativo, dado que suelen ser notoriamente más pequeñas que las poblaciones urbanas y reportan una menor actividad contaminante que éstas últimas. Generalmente este tipo de sesgo es el

resultado de una conveniencia, en donde los sujetos son el único grupo disponible y tienden a ser de una categoría demográfica estrecha. Si bien los resultados son válidos como tales, el desafío está en advertir el sesgo y que los investigadores sean conscientes de que no pueden extrapolar los resultados a toda la población.
En nuestro caso de estudio, notamos que ciertas proteínas o familias de proteínas aparecen en *PDB* mucho más frecuentemente que otras. Pensamos que la razón de esto está relacionada con el hecho de que resultan de mayor interés, lo cual puede deberse a varios motivos: porque tienen un mayor valor biológico y por ello se ha buscado estudiarlas en profundidad; o bien porque forman parte de algún circuito clave del organismo; porque resulta más fácil experimentar con ellas que con otras; o debido a que sirven a modo de comparación (en el caso de probar la misma droga con dos proteínas similares). Sea la razón que fuere, nos encontramos bajo un caso que podríamos catalogar como *sesgo de inclusión*, es decir, un "condicionamiento" de los datos. Esto puede derivar en un problema para las técnicas de aprendizaje automático dado que, al inferir patrones a partir de unas pocas muestras, las predicciones de los modelos se verán cuanto menos comprometidas.
Para hacer frente a este inconveniente, el grupo de la BIA aplicó filtros para "balancear" el dataset, haciendo que queden representadas una cantidad balanceada de secuencias por familia.

2.1.1.1 Filtro por *Pfams* (redundancia de familias)

Se realizó un agrupamiento por familias, según los dominios *Pfam* de cada proteína. No se usó un programa específico, sino que se armó un script basado en `hmmer`[41], un programa que suele usarse para análisis de secuencias en bases de datos de dominios. Se utilizó un *cut-off* de 3 proteínas por familia como máximo, buscando eliminar la sobre-representación de secuencias similares.

2.1.1.2 Filtro por identidad

Cuanto mayor sea la identidad entre secuencias, más similar se espera que sea su estructura, por lo tanto, se hipotetiza que los pockets también serán similares. Por ello, se aplicó un filtro por identidad, el cual se efectúa mediante alineamiento de secuencias de aminoácidos de las proteínas. Se acordó agrupar proteínas con un 90% o más de identidad de secuencia usando el programa `cd-hit`[42,43], que genera un nuevo conjunto de proteínas, agrupado por el *cut-off* de identidad que se indique. De esta forma, quedaron seleccionados unos pocos representantes por cada grupo.

2.1.2 Filtro de cofactores y cosolventes

Varios de los experimentos de interacción que figuran en las bases de datos (e.g., PDB) involucran lo que se conoce como cofactores y cosolventes. Un cosolvente se define como "*una sustancia química utilizada en pequeñas cantidades para mejorar la efectividad de un solvente primario en un proceso químico*", mientras que un cofactor es "*un componente no proteico, termoestable y de baja masa molecular, necesario para la acción de una enzima*". En ambos casos son moléculas que a nuestros fines no resultan de utilidad.
Se realizó un filtrado de los experimentos que involucraban dichas moléculas[c], lo cual redujo

[c] Para ello se dispone de un archivo con una lista de cofactores y cosolventes conocidos. Este archivo puede encontrarse en el Apéndice C (Material Suplementario).

los datos totales disponibles. En la sección siguiente se describen los números de registros del dataset resultante.

2.1.3 Características del dataset curado

Finalmente, el dataset preparado por el BIA contiene 19 descriptores para los pockets y 11 para los ligandos. La cantidad de registros en cada uno de los archivos puede verse en la tabla 2.3.

Archivo	Cantidad de registros
pockets.csv	402.795
ligands.csv	122.095
bindings.csv	345.314

Table 2.3: Cantidad de registros de los archivos elaborados por el BIA.

2.2 *Data Enrichment*

El concepto de enriquecimiento de datos (en inglés, *data enrichment*) se refiere al proceso de refinar y mejorar los datos crudos recolectados. Como primer paso para llevar a cabo esta tarea, realizamos un extenso trabajo de análisis exploratorio de los datos, a fin de comprender mejor las características de nuestro dataset. Para ello, utilizamos métodos de estadística descriptiva para tener un panorama general, con el cual podamos formular las primeras hipótesis. La puesta a prueba de estas hipótesis no sólo puede dar inicio al desarrollo de modelos de aprendizaje automático, sino también a nuevas hipótesis que a su vez nos pueden invitar a la revisión del conjunto de datos, y aplicar transformaciones sobre éste. En otras palabras, podemos decir que este análisis exploratorio fue iterativo y continuo.
Para todo este desarrollo utilizamos código en lenguaje Python, en forma de notebooks de *Jupyter* [44]. Ésta es una aplicación web interactiva para cómputo, que permite crear documentos que combinan ejecución de código con texto formateado, gráficos, imágenes y tablas, entre otros. Las notebooks se encuentran disponibles como parte del material suplementario (ver Appendix C).

Pre-procesamiento: valores nulos

Como primer medida realizamos un proceso de limpieza, con el fin de identificar datos inválidos o incompletos, y, en caso de existir, corregirlos o eliminarlos. Un dato correspondiente a una variable puede, por ejemplo, tener valor nulo debido a que no existen en la base de datos consultada. O bien, particularmente en el caso de variables numéricas, tener como valor `NaN` (*Not A Number*), lo cual puede indicar problemas a la hora de calcular su valor, ya sea por razones numéricas, de hardware o causas desconocidas. Inclusive puede haber errores por fuera de la etapa de cálculo, en la etapa de escritura de los valores al archivo.
El análisis realizado dejó en claro que no existe ningún problema con las propiedades de pockets, pero sí con las de los ligandos, dado que algunas propiedades presentan un valor definido para muy pocas instancias (para ver en detalle cuáles, consultar el Apéndice A, sección A.2.5.2). Dada la magnitud del caso, intentar obtener la información de otra fuente podría resultar en un procesamiento costoso, sin garantía de que no se encuentren problemas similares una vez finalizado. Por este motivo, optamos directamente por eliminar dichas propiedades del dataset.

2.2.1 Unificación de pockets duplicados

Con el fin de tener un mejor entendimiento de la distribución que tenían los pockets, prestamos atención a cuánto se parecen entre sí en función de los valores de sus propiedades. Analizando si los identificadores de pocket se repetían, observamos que en ningún caso figuraba que un mismo pocket se uniese a más de un ligando. Esto, en principio, no parecía corresponderse con lo que la bibliografía sugería[45]. Estudiando en detalle encontramos que muchos registros de pockets tenían todos sus features idénticos (ver tabla 2.4).

En consecuencia, optamos por definir un `unified_pocket_id` utilizando el vector de features como identificador del pocket[a] (ver tabla 2.5). Almacenamos los pockets "unificados" en el archivo `unified_pockets.csv`, y preservamos las correspondencias entre cada entrada de este archivo (usando el nuevo identificador) y los pares proteína-pocket originales en un nuevo archivo llamado `annotated_pockets.csv` (ver tabla 2.6).
Una vez hecha esta unificaciónlos logramos que un mismo `unified_pocket` se una a más de un ligando. Analizaremos más adelante las características de dichas interacciones.

[a] A nivel implementativo, optamos por un hash de este vector a fin de facilitar las operaciones (obviamente, chequeando que no hubiesen colisiones).

Table 2.4: Ejemplo de entradas de `pockets.csv` con pockets duplicados.

pdbID	pocketID	feature 1	...	feature k
2ICZ	0	**2.1111**	...	**0.91531**
2ICZ	1	**2.1111**	...	**0.91531**
2ICZ	2	**2.1111**	...	**0.91531**

Table 2.5: Unificación de los pockets de la tabla 2.4 incorporando un identificador único (hash id).

unified_pocket_id	feature 1	...	feature k
hash(features 1..k)	**2.1111**		**0.91531**

Table 2.6: Ejemplo de entradas del archivo `annotated_pockets.csv`

pdb_id	pocket_id	unified_pocket_id
2ICZ	1	1253797820612730549
2ICZ	2	5674623373931742224

2.2.2 Análisis de bindings

Una vez definido el `unified_pocket_id`, nos abocamos a comprender la correspondencia entre pockets y ligandos, analizando las interacciones que figuran en el archivo `bindings.csv`. En particular nos preguntamos si al agrupar los pockets usando alguna medida de similitud, los ligandos unidos a ellos también se agrupan de una manera coherente. De esta forma estaríamos reforzando la idea de que existe una relación entre los descriptores que elegimos para representar a pockets y ligandos.
Para responder a esta pregunta utilizamos herramientas de **agrupamiento** (en inglés, *clustering*) provistas por la biblioteca `scikit-learn`[46], focalizándonos en grupos de datos "parecidos". La manera de agrupamiento de las muestras por lo general se hace bajo criterios que involucran alguna medida de similitud, como puede ser la distancia euclediana en caso de tratarse de vectores numéricos. Una de las técnicas de clustering más conocidas es K-means[d], y fue la que decidimos utilizar para esta etapa. La técnica toma como parámetro la cantidad de clusters k en los cuales se quiere agrupar los elementos del conjunto (pockets o ligandos en nuestro caso). Utilizando esta técnica, realizamos el siguiente experimento: hicimos múltiples corridas

[d] La biblioteca sci-kit learn implementa K-means de una manera muy simple de utilizar. Una descripción más detallada de su algoritmo puede encontrarse en el Apéndice A, sección A.2.8.

de clustering sobre los pockets incrementando el valor de k. A continuación, para cada corrida obtuvimos el identificador del *cluster* (al cual de ahora en más denominaremos *etiqueta*) de cada pocket y asignamos esa misma *etiqueta* a los ligandos que figuraban con interacciones con dicho pocket en el archivo `bindings.csv`. A partir de esto, se define naturalmente un agrupamiento de ligandos, utilizando como medida de similitud aquellos que tengan la misma *etiqueta*. Finalmente, medimos qué tan bueno era el clustering de pockets y el clustering asociado de ligandos y analizamos cómo variaba en función de k. Resulta evidente que cuando k es igual a la cantidad de elementos (pockets), obtenemos k cluster con un solo elemento cada uno y, por lo tanto, la cohesión de cada grupo es máxima. En el caso opuesto, cuando $k = 1$, la cohesión es mínima ya que cuenta con un solo grupo con la totalidad de elementos (algunos muy disímiles entre sí).
Existen diversas métricas para determinar qué tan bueno es un clustering y así poder comparar uno con otro. Para nuestro caso particular nos enfocamos en medir la **inercia** de cada cluster. La inercia de un cluster se define como la suma de las distancias cuadradas de cada muestra al centro del cluster. Es una medida de cohesión interna del cluster, presentando valores pequeños cuando los puntos se encuentran muy cerca del centro y valores muy grandes cuando se encuentran lejos. En nuestro caso, es deseable que la inercia vaya decreciendo en función del valor k, dado que esto podría interpretarse como que los pockets/ligandos se parecen mucho entre ellos.
A continuación presentamos los resultados de este análisis y describimos cómo construímos las imágenes para representarlos gráficamente.

2.2.2.1 Bolsillos a ligandos

Para representar los conceptos detallados anteriormente, optamos por definir un BubbleChart que muestre los valores de la inercia de los clusters en función de k, y en donde el valor del tamaño de las burbujas se corresponden con los tamaños de cada cluster (cardinal del conjunto), promediados por el total de sus elementos.
Al generar este gráfico, en primer lugar, llevamos a cabo una normalización de los datos tanto de pockets como de ligandos mediante el método `normalize` de `sci-kit learn`, que reescala cada vector para que cada muestra tenga norma unitaria. Esto es necesario dado que en caso contrario las diferencias que pueden haber en los valores de las inercias alteran severamente la legibilidad del gráfico y no permiten realizar una buena interpretación. Luego, para cada corrida de K-means (es decir, cada uno de los k), calculamos la inercia de cada cluster de pockets y ligandos, promediada con el número total de sus elementos. Por último, dado que buscamos mostrar los resultados de los agrupamientos de pockets y ligandos en el mismo gráfico, para el caso de los pockets, invertimos los valores de inercia de los clusters (simplemente multiplicando por -1), de forma que apareciesen en la mitad inferior de la imagen. Esto puede verse en la Figure 2.1, en donde variamos k de 2 a 100.

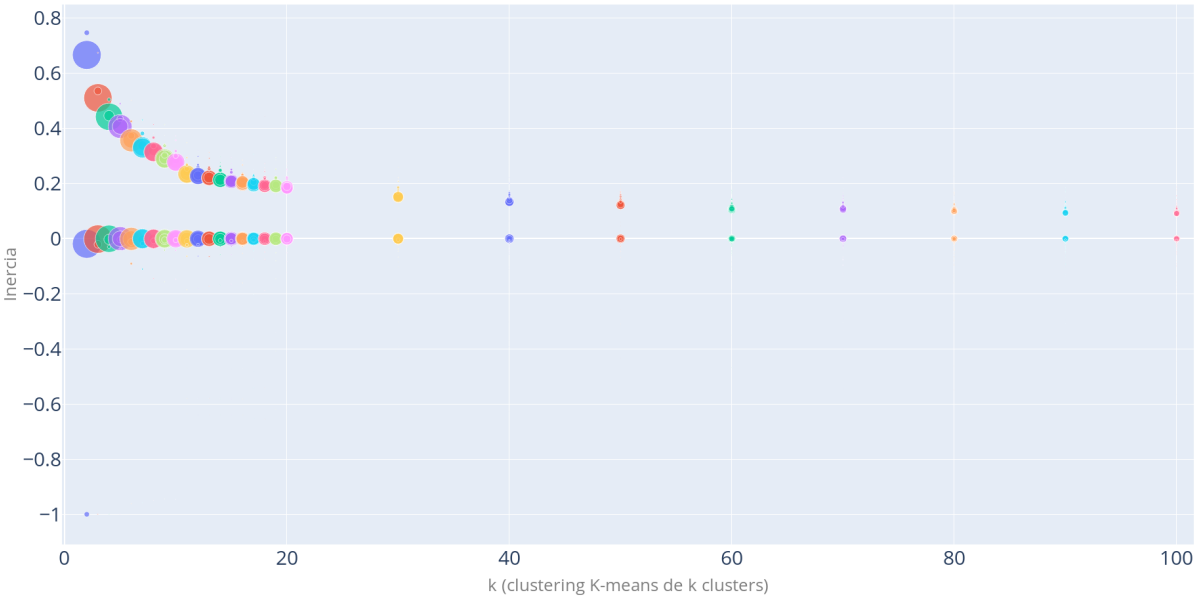


Figure 2.1: Bubblechart de clustering de pockets variando de 2 a 100 el número de clusters k usado como parámetro de K-means. En la mitad inferior se visualizan los clusterings de pockets y en la mitad superior los de sus ligandos asociados.

En el segundo gráfico (Figure 2.2) optamos por un ScatterPlot en donde cada punto representa un cluster. Nos inclinamos por esta alternativa en lugar de la anterior debido a que buscábamos hacer un clustering más extenso (aumentan el valor de k hasta 1.000), y a medida que el valor de k crece, comienzan a aparecer clusters muy pequeños en tamaño, los cuales prácticamente son indistinguibles en el BubbleChart de la figura 2.1.

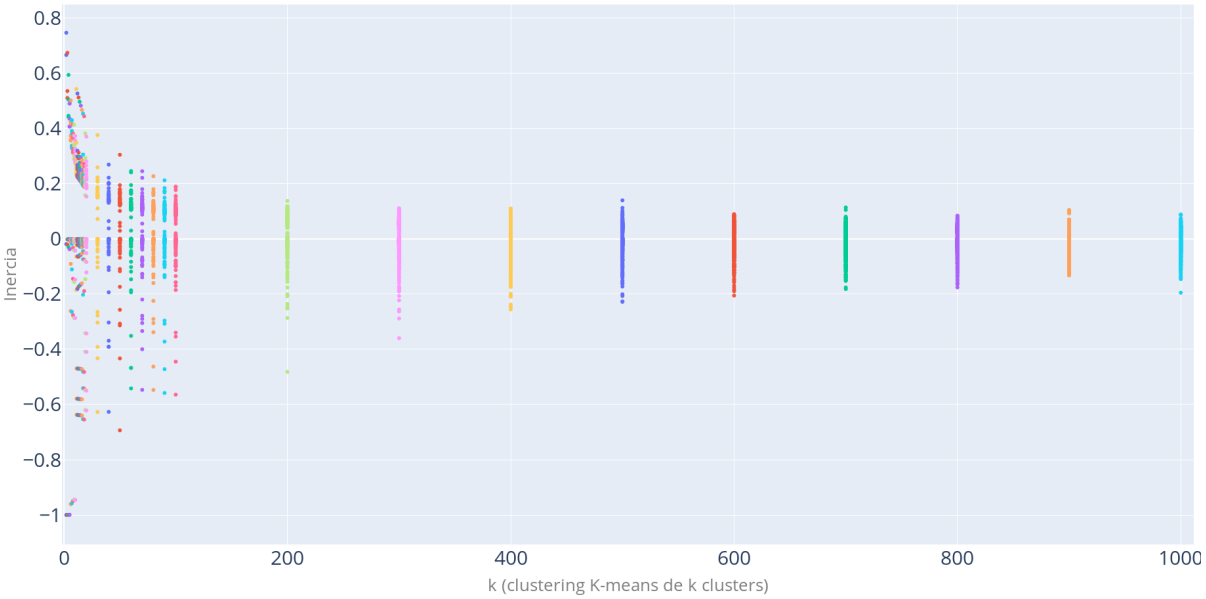


Figure 2.2: Scatterplot variando el número de clusters k de K-means de 2 a 1.000. Cada punto representa un cluster. En la mitad inferior se visualizan los clusterings de pockets y en la mitad superior los de sus ligandos asociados.

Ambas figuras muestran que a medida que los pockets se parecen más entre sí (la inercia se acerca a 0), los ligandos asociados también se parecen mucho entre sí (mismo comportamiento), lo cual refuerza la hipótesis de que existe cierta correlación entre ambos en función de las propiedades elegidas. Trabajos previos[47] se refieren a este comportamiento como un fenómeno de "culpa por asociación" y es tomado como un buen puntapié inicial para inspirar técnicas de recomendación de pockets a ligandos.
Por otro lado, los pockets presentan un comportamiento muy particular: se agrupan muy bien desde el primer momento. Si bien el ScatterPlot de la Figure 2.2 no parece tan ordenado, el BubbleChart de la Figure 2.1 muestra que los pockets de mayor tamaño (y por ende los más representativos del conjunto de datos) sí muestran esta tendencia. No hemos profundizado en la causa de dicho comportamiento, pero muy probablemente esté relacionada al hecho de que los pockets de la misma proteína o de una familia de proteínas posean propiedades muy similares.

2.2.2.2 Ligandos a bolsillos

De la misma forma que en la sección anterior, planteamos un análsis de bindings para el problema de *Inverse Virtual Screening*. Los gráficos fueron generados de manera análoga, tomando como punto de partida los ligandos que tienen alguna entrada en el archivo `bindings.csv` con pockets, viendo qué tan bueno es el agrupamiento de los pockets con los cuales estos ligandos tienen interacciones. Los resultados pueden visualizarse en las figuras 2.3 y 2.4.

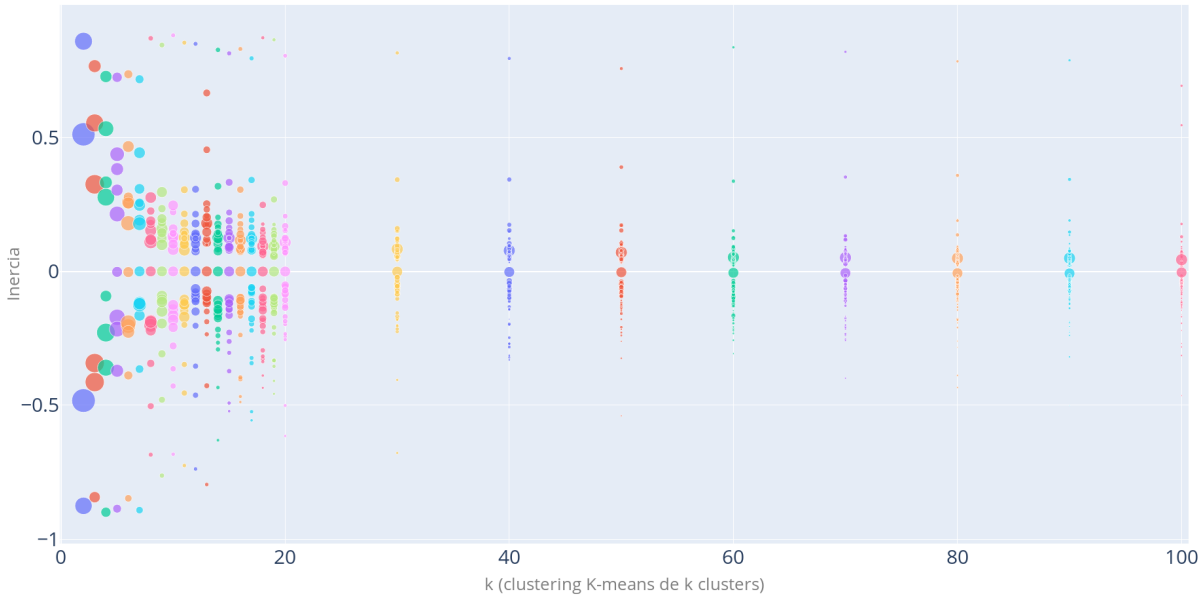


Figure 2.3: Bubblechart de clustering de pockets variando de 2 a 100 el número de clusters k usado como parámetro de K-means. En la mitad inferior se visuaiza el clustering de ligandos mientras que en la mitad superior se ubican los clusters de los pockets asociados.

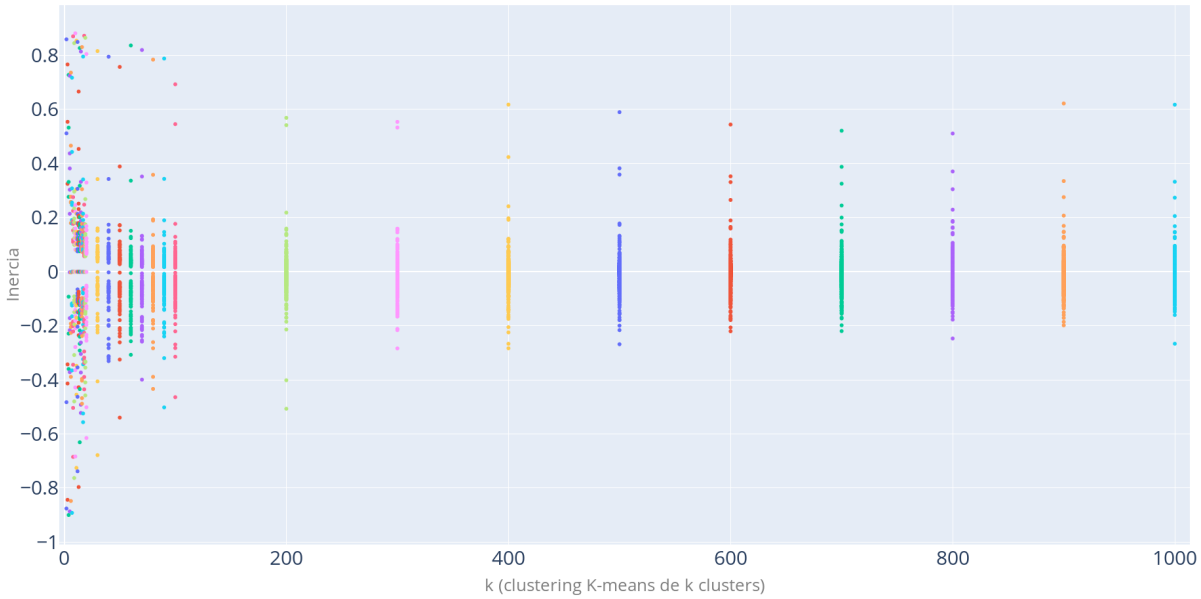


Figure 2.4: Scatterplot variando el número de clusters k de K-means de 2 a 1.000. Cada punto representa un cluster. En la mitad inferior se visuaiza el clustering de ligandos mientras que en la mitad superior se ubican los clusters de los pockets asociados.

A diferencia del caso pocket-a-ligando mencionado anteriormente, la figura 2.3 muestra en principio un comportamiento más caótico. En particular los ligandos no se agrupan de forma tan homogénea entre sí (parte inferior del gráfico 2.3), como sí lo hacían los pockets entre ellos en la figura 2.1. Por otro lado, si bien ambos ScatterPlots (figuras 2.2 y 2.4) mejoran los valores de inercia a medida que aumenta el valor de k, en la figura 2.4 se ve que la inercia de los clusters

de pockets proyectados tiene oscilaciones y no parece converger a valores cada vez menores. Nuestra hipótesis acerca de este comportamiento es que la cantidad de ligandos con los que contamos son demasiado pocos, dado que aproximadamente sólo 2.500 presentan interacciones en `bindings.csv`. Este planteo devino finalmente en una nueva modificación del dataset mediante la inclusión de más registros a partir de nuevas fuentes de datos. En la sección 2.2.7 volveremos sobre este tema para plantear alternativas que permitan mejorar el dataset asociado al problema de Inverse Virtual Screening.

2.2.2.3 Distribución de bindings

Continuando con el análisis de las interacciones entre pockets y ligandos, nos postulamos el siguiente interrogante: ¿cuántos ligandos se unen a cada pocket? Y de forma análoga: ¿cuántos pockets interactúan con un determinado ligando? Es decir, estudiar la forma en que se distribuyen las interacciones que figuran en el archivo `bindings.csv` en función pockets y de ligandos.

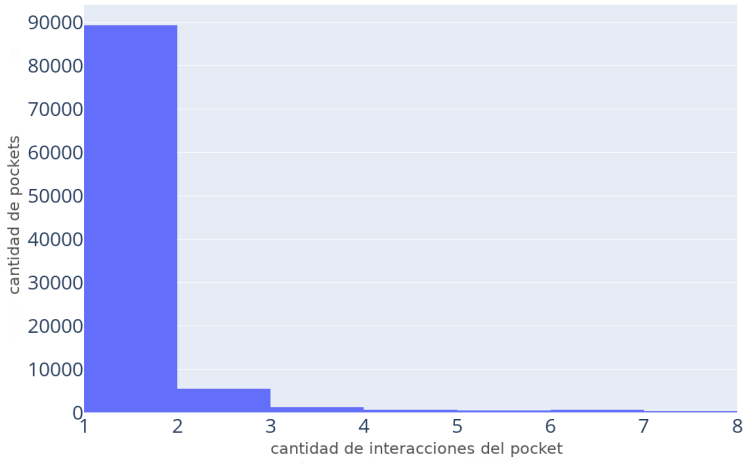


Figure 2.5: Frecuencia de pockets en función de la cantidad de ligandos con los que interactúan.

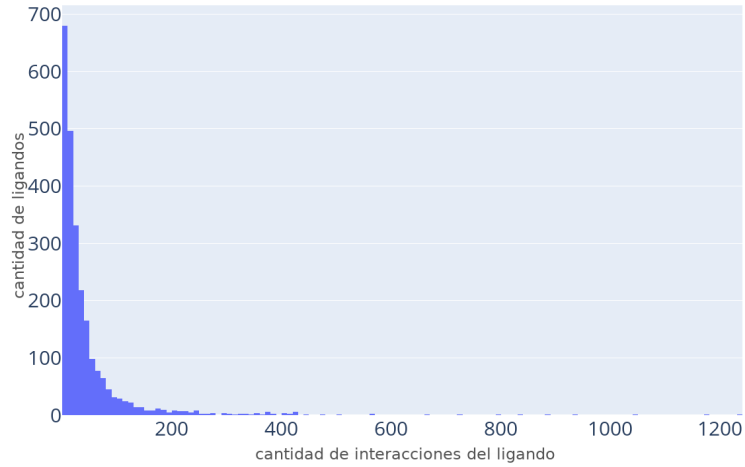


Figure 2.6: Frecuencia de ligandos en función de la cantidad de pockets con los que interactúan.

Para dar respuesta a estas preguntas, optamos por construir histogramas con la frecuencia de aparición de pockets/ligandos en función de la cantidad de interacciones (figuras 2.5 y 2.6 respectivamente). En la figura 2.5 puede verse la frecuencia de la cantidad de uniones de pockets a ligandos. Se evidencia que la gran mayoría de pockets figura en una única interacción. Las razones de esto posiblemente estén relacionadas a sesgos experimentales (fenómeno descripto anteriormente en este capítulo, sección 2.1.1).

En la figura 2.6 puede apreciarse un fenómeno similar al de la figura 2.5: la mayor parte de los ligandos figuran en muy pocas interacciones.
Ambos gráficos son muy similares, lo cual refuerza la idea de un sesgo experimental existente en las bases de datos consultadas. La diferencia más notable entre ambos gráficos es la "suavidad" de la curva, la cual está relacionado a que los ligandos aparecen en interacciones con mayor versatilidad que los pockets.

2.2.2.4 Consolidación de pockets: pocketoids

Para el caso específico del problema de "esparcimiento" o de la versatilidad de los pockets detallado en la sección anterior, hicimos pruebas con algunas variantes que pudiesen aportar más información de este fenómeno. En este punto, contábamos con una asunción del dominio del BIA acerca de que los pockets probablemente tuviesen valores de sus descriptores muy similares entre ellos. Esa asunción fue en cierta forma validada por la figura 2.1, en donde los clusters de pockets rápidamente adquieren una inercia baja. Es decir, la presencia de mínimas diferencias en los valores de las features de pockets no llegan a alterar la viabilidad de las interacciones con sus ligandos. En función de esto, exploramos la utilización de K-means para poder describir de forma sintética la totalidad de los pockets. Esta descripción sintética se consigue sustituyendo la descripción de todos los elementos de un cluster por la de un representante característico del mismo, que usualmente es el centro del cluster, también denominado "centroide".

A partir de esto, realizamos un clustering sobre los pockets variando el parámetro k, con el fin de obtener conjuntos de pockets muy parecidos entre sí. Utilizamos el centroide del cluster y definimos un *pocketoide*, un pocket "imaginario" cuyo conjunto de propiedades describen en sí el cluster. Modificamos las interacciones para que todos los ligandos que se vinculen a un pocket del cluster pasen a estar unidos a un pocket *ideal* cuyas features son las del *pocketoid* de dicho cluster. La nueva distribución de bindings puede verse en la figura 2.7 para un clustering con un valor de k de 1.000.

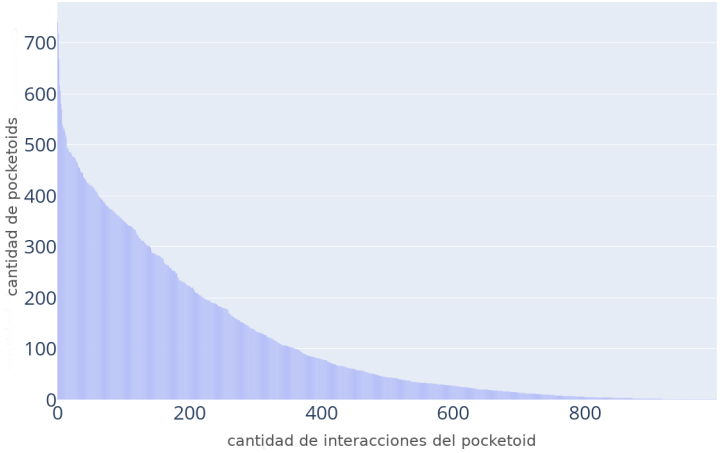


Figure 2.7: Frecuencia de pocketoids (calculados mediante un clustering de k =1.000) en función de la cantidad de ligandos con los que interactúan.

Se puede observar una curva "suave" en comparación al gráfico escalonado de la figura 2.5. Este resultado puede resultar poco relevante por sí solo, pero más adelante cuando mencionemos las distintas alternativas para implementar sistemas de recomendación, resultará evidente que la distribución de los datos puede causar serios problemas en la performance de algunos algoritmos. Por cuestiones de tiempo no indagamos más en este experimento, y tampoco lo utilizamos para entrenar nuestros modelos de aprendizaje automático. En el capítulo 8 (Trabajo Futuro), detalleremos sin embargo algunas cuestiones interesantes con respecto a la utilización de este enfoque para nuevos desarrollos.

2.2.3 Distribución de los datos y tests de normalidad

Teniendo en cuenta los resultados de los análisis anteriores, buscamos hallar una correlación entre los descriptores de pockets y ligandos. Es decir, por ejemplo, si existe una correlación entre el descriptor `mwt` de ligandos y el descriptor `volume` de pockets.
Existen varias técnicas disponibles para calcular coeficientes de correlación. Una de los más conocidas es el coeficiente de Pearson[48], el cual define una medida lineal entre dos variables. El cálculo de dicho coeficiente requiere que las variables de los descriptores cuenten con una distribución normal, y, si bien los valores con los que contamos fueron obtenidos de experimentos provenientes del mundo real (por lo cual esperamos que naturalmente sigan una distribución normal), realizamos histogramas y, a nivel más formal, **tests de normalidad**[49], a fin de validar esta hipótesis.
Un histograma de frecuencia es un método de inspección visual que puede resultar útil para formular las primeras hipótesis sobre la distribución de la cual provienen los datos. Realizamos un histograma por cada una de las features de pockets y ligandos, los cuales se encuentran todos detallados en el Apéndice C de Material Suplementario. En ellos se presenta no sólo el histograma en sí, sino que se detalla información referente a la curva normal que debería darse en función de la esperanza y el desvío estándar de la distribución. Como ejemplo, en la figura 2.8 se ve el caso de la propiedad de pockets `alpha_sphere_density`. En este caso, no parece haber

una marcada concordancia con una distribución normal. Por otro lado, en un caso diferente, el de la propiedad `hydrophobicity_score` (ver figura 2.9), visualmente parece mostrarse una clara distribución normal.

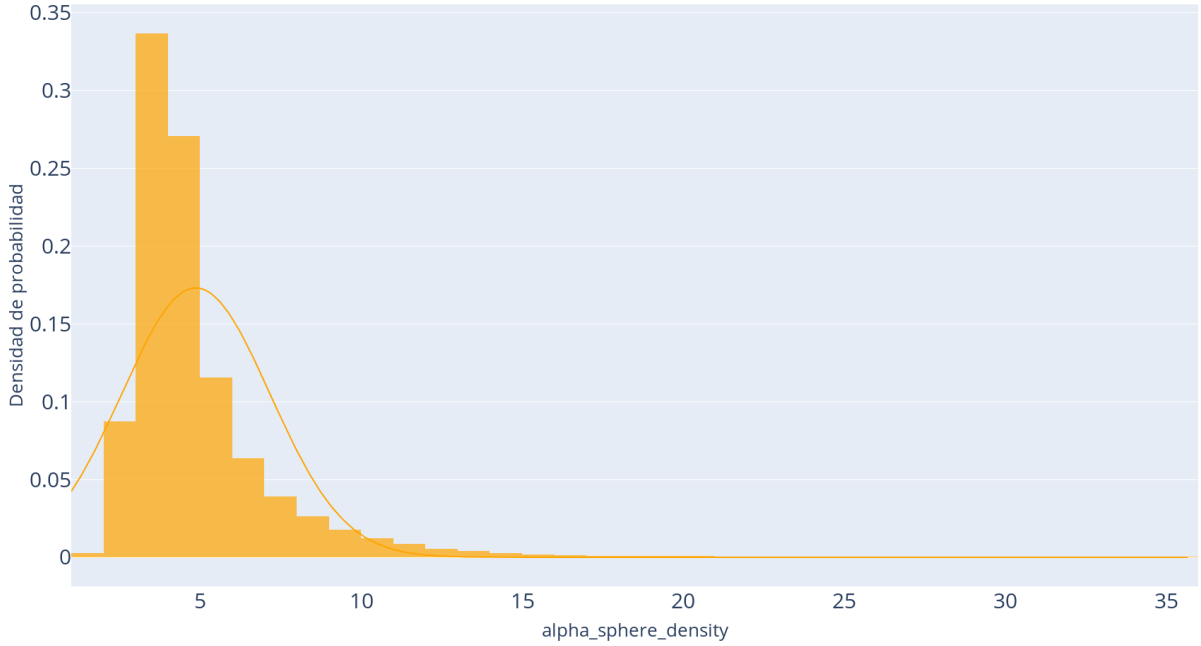


Figure 2.8: Distribución de `alpha_sphere_density` (archivo `unified_pocket_id.csv`) combinado con una curva normal basada en la esperanza y el desvío estándar de la distribución.

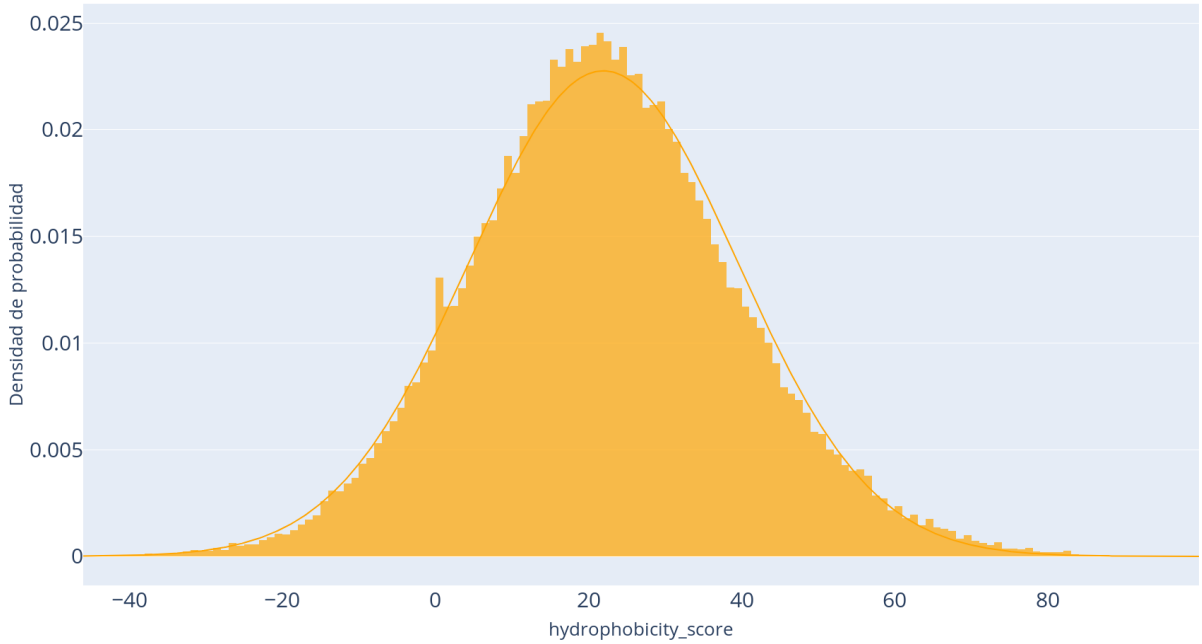


Figure 2.9: Distribución de `hydrophobicity_score` (archivo `unified_pocket_id.csv`) combinado con una curva normal basada en la esperanza y el desvío estándar de la distribución.

La desventaja del histograma, al igual que con otros métodos visuales (*Q-Q plots*, *P-P plots*, diagramas de caja), es que no brinda ninguna garantía real de que los datos sigan una distribución normal. Por eso es que decidimos complementarlos con tests de normalidad provistos por la bilbioteca `sci-kit learn` [e], que comparan los valores de la muestra (e.g., valores de un descriptor contra un conjunto de valores de una distribución normal de misma media y desvío estándar). En éstos, se propone una **hipótesis nula**, que plantea que la distribución de la muestra es normal y, si el test es significativo (*p-valor* menor que un α dado, usualmente 0.05), puede rechazarse dicha hipótesis y concluirse efectivamente que la distribución no es normal. Para nuestra sorpresa, los tests aseguran la no-normalidad para todos los casos.
La aplicación de dichos tests requiere ciertos recaudos. Por ejemplo, para tamaños de muestra pequeños, los tests tienen poco poder para rechazar la hipótesis nula. Mientras que para tamaños grandes, se vuelven sensibles a desviaciones pequeñas[f]. Esto último puede verse en la figura 2.10.

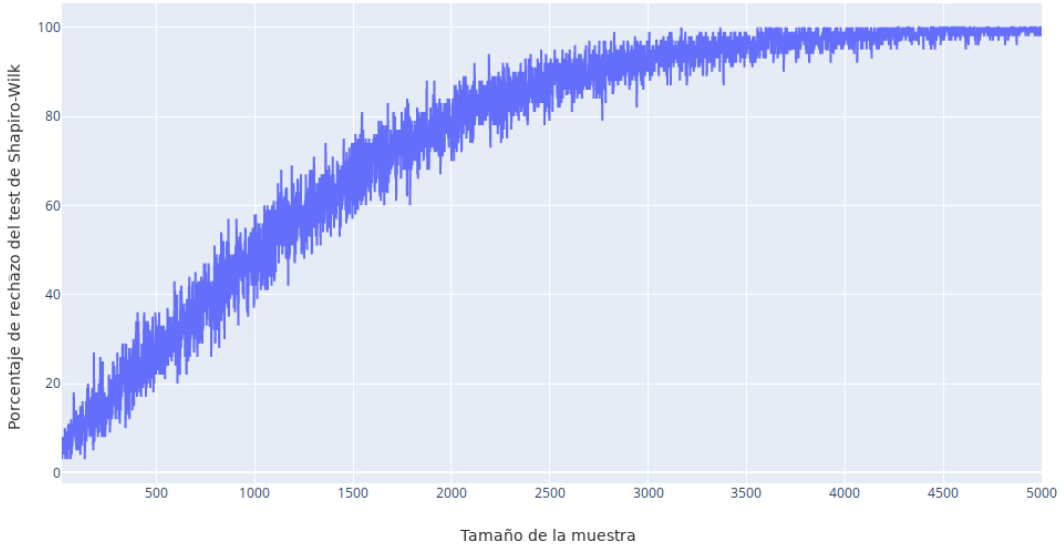


Figure 2.10: Porcentaje de rechazo del test de Shapiro-Wilk (p-valor < 0.05) de un total de 100 muestras aleatorias, variando el tamaño de muestra. Los datos corresponden a la feature `hydrophobicity_score` (archivo `unified_pocket_id.csv`) cuya distribución de datos se puede visualizar en la Figure 2.9

Teniendo esto en consideración, puede explicarse que los tests de normalidad fallasen debido a los tamaños de las muestras con los que contamos, no permitiendo validar que las propiedades siguiesen una distribución normal. Debido entonces a que no podemos utilizar métodos automáticos para comprobar que los datos siguen una distribución normal, no podemos asegurar que se cumplan las condiciones para utilizar el método de Pearson. Por este motivo decidimos optar por otras alternativas, las cuales detallamos a continuación.

[e] Utilizamos la función `normaltest` implementada en la biblioteca Scipy [50], basada en el test de D'Agostino-Pearson [51,52].

[f] En todos los test de normalidad, la medida *kurtosis* se vuelve altamente sensible, de forma que una diferencia muy sutil en los valores de la muestra puede tener impacto notable a medida que aumenta el tamaño de la muestra [53].

2.2.4 Correlación entre features

Como ya mencionamos, para el cálculo de correlación, contamos con el coeficiente de correlación de Pearson, que mide el grado en que dos variables aleatorias continuas y normalmente distribuidas se correlacionan linealmente. Por otro lado, contamos también con el coeficiente de correlación de Spearman, que, a diferencia del de Pearson, se calcula entre dos variables aleatorias que pueden ser tanto continuas como discretas y sin asunción sobre la distribución de ambas. Es una medida de la monotonicidad de la relación entre ambas variables. Dicho de otro modo, a medida que $\mathbb{X}$ crece, $\mathbb{Y}$ crece; o bien, a medida que $\mathbb{X}$ crece, $\mathbb{Y}$ decrece. También puede considerarse un caso particular del coeficiente de correlación de Pearson, calculado sobre los rankings de las variables[g], por lo que es más robusto ante la presencia de outliers[48][h].
A partir del método de Spearman calculamos la correlación entre cada una de las propiedades, y el resultado fue graficado por medio de un **mapa de calor** (en inglés, *heatmap*), el cual puede verse en la figura 2.11. En éste se encuentran volcados los coeficientes de correlación para todos los pares de propiedades de pocket, los de ligandos y ambos una vez combinados, a través del archivo `bindings.csv`. Los valores se rigen por la escala de colores que aparece a la derecha, siendo los más significativos aquellos cuyo valor absoluto es mayor. El gráfico es simétrico dado que aparecen las mismas propiedades en ambos ejes. Esta es también la razón por la cual en la diagonal puedan encontrarse celdas de máxima correlación, dado que es la correlación de una propiedad consigo misma. En el gráfico se evidencian algunas correlaciones fuertes y muchas otras más bien débiles. Los umbrales para determinar si una correlación es fuerte o no pueden variar de acuerdo al área de estudio, y son actualmente motivo de debate[54].

[g] Se reemplazan cada valor de la muestra por su posición en una secuencia ordenada.
[h] En el Apéndice A, sección A.2.9 se realizan algunas comparaciones entre los distintos métodos.

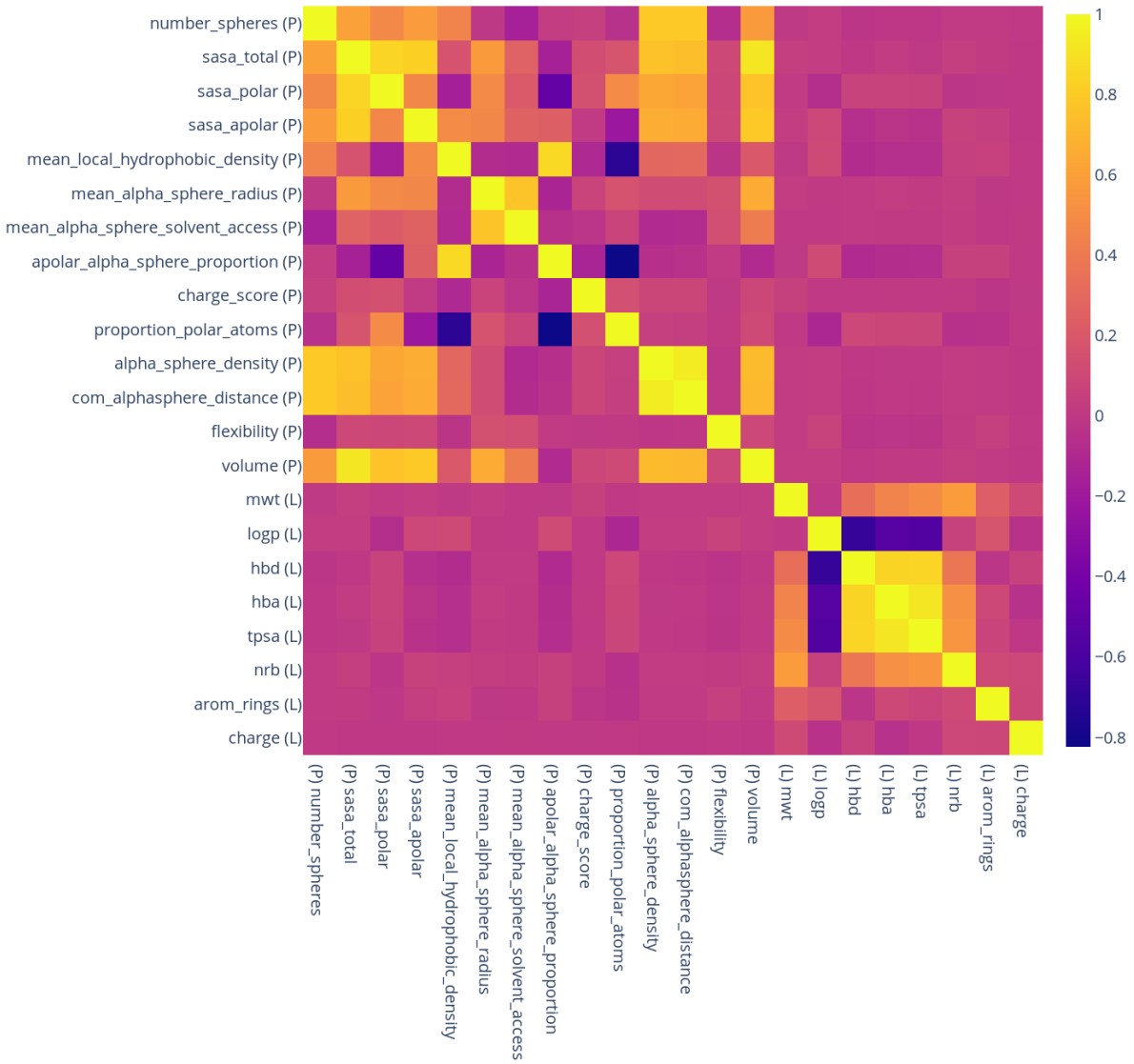


Figure 2.11: Heatmap del coeficiente de correlación de Spearman correspondiente a las propiedades de `pockets.csv` y `ligands.csv`, asociadas a través de `bindings.csv`. Las propiedades que figuran con `(P)` son las de pockets y con `(L)` las de ligandos.

En la figura 2.11 se observa que las mayores correlaciones se dan entre features de pockets por un lado y ligandos por otro. La correlación entre pockets y ligandos es marcadamente menor. Un resultado muy llamativo hubiese sido que exista algún par de propiedades de pockets y ligandos que tengan una correlación muy fuerte, dado que hubiese sido un dato muy relevante en el diseño de un sistema de recomendación de pocket a ligando y viceversa: sencillamente usando ese par de variables podría determinarse unívocamente si el acoplamiento entre ambos es factible o no. Este comportamiento sin embargo, no se evidencia en el gráfico. Si bien existen pares de propiedades con índices de correlación más altos que otros, lo más llamativo del gráfico es que la correlación entre propiedades de pockets y propiedades de ligandos es baja. Para medir qué tan baja es, realizamos el mismo test, pero con un dataset que contiene los mismos datos pero con las interacciones presentes en `bindings.csv` "desordenadas". Para desordenar los datos del

dataset original corrimos el método `shuffle` de `scikit-learn` sobre las relaciones existentes entre pockets y ligandos del archivo de bindings. El resultado puede verse en la figura 2.12.

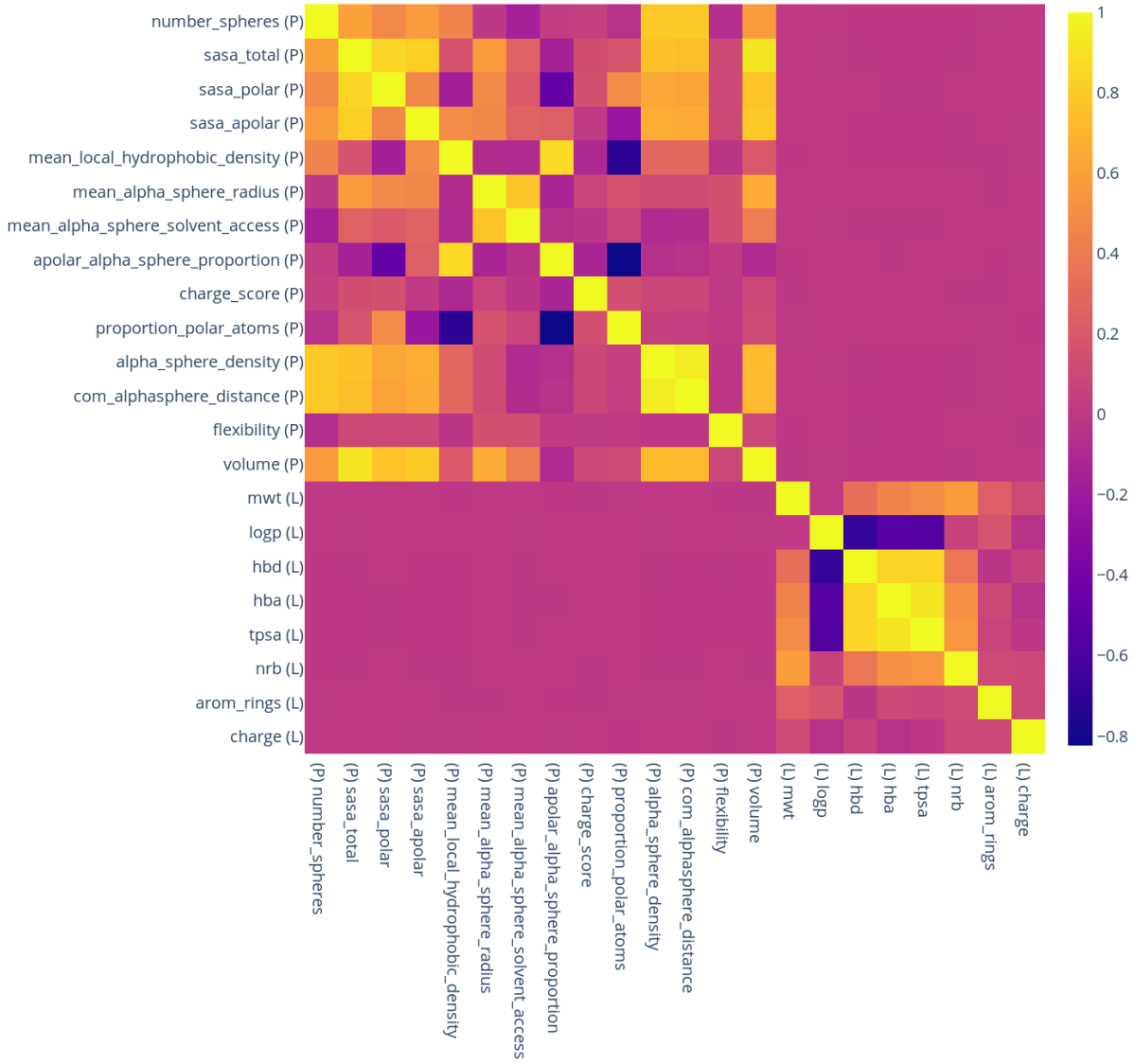


Figure 2.12: Heatmap del coeficiente de correlación de Spearman correspondiente a las features de `pockets.csv` y `ligands.csv`, asociadas a través de una versión desordenada de `bindings.csv` Las propiedades que figuran con `(P)` son las de pockets y con `(L)` las de ligandos.

Como se puede observar en la figura 2.12, los niveles de correlación de este experimento son notoriamente más bajos que la versión original para todo par `propiedad de pocket, propiedad de ligando`. Con esta prueba podemos evidenciar que, si bien la correlación original es baja, sigue aportando algún tipo de información de la relación entre los conjuntos de propiedades de pockets y ligandos.
Para contraponer estas medidas, optamos por realizar una comparación valor a valor de cada cruce de variables. Para ello generamos un gráfico que consiste en dos histogramas a partir de

los valores de cada "cuadrado" de la zona `pocket-ligando` del heatmap original y del heatmap desordenado (*shuffled*). Los coeficientes de correlación de la figura 2.12 se concentran cerca del valor 0.

Por otro lado, los valores de la figura 2.11 se distribuyen de una manera más amplia, como puede verse en la figura 2.13.
Otra forma de ver lo mismo es mediante un scatterplot (ver figura 2.14), asignando al eje x cada uno de los pares `(propiedad de pocket, propiedad de ligando)` y al eje y su valor de correlación obtenidos de las figuras 2.12 (color azul) y 2.11 (color rojo).
En ambas figuras se evidencia una clara diferencia entre la versión con los datos originales y la versión con los datos desordenados. En la versión desordenada la correlación es prácticamente nula para cualquier combinación de propiedades de pockets y ligandos.
Por último, a fin ampliar el análisis de los datos, graficamos un heatmap de la correlación de Spearman extendido al conjunto de ligandos totales que figuran en el archivo `ligands.csv`, dado que muchos no se encuentran en la figura 2.11 por no tener interacciones con ninguno de los pockets[i]. El resultado puede observarse en la figura 2.15, en donde puede notarse que los ligandos ampliados, es decir, no solamente los que tienen interacciones con pockets (guardados de ahora en más en el archivo `reduced_ligands.csv`) muestran una ligera diferencia: las features: `hba` y `hbd` no presentan una fuerte correlación ni entre ellas ni con `tpsa`.

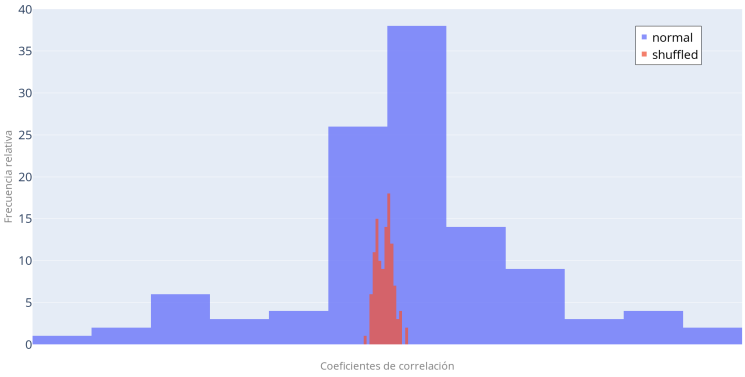


Figure 2.13: Frecuencia relativa de los valores de la región pocket-ligando de la matriz de correlación de Spearman. Cada elemento del eje X representa un par [feature de pocket, feature de ligando]. En azul, datos de la figura 2.12 (datos originales). En rojo, datos de la figura 2.11 (datos desordenados).

Figure 2.14: Scatterplot de los valores de la región pocket-ligando de la matriz de correlación de Spearman. Cada elemento del eje X representa un par [feature de pocket, feature de ligando]. En azul, datos de la figura 2.12 (datos originales). En rojo, datos de la figura 2.11 (datos desordenados).

[i] Cabe aclarar que, para el caso de pockets, este fenómeno no ocurre debido a que, a partir de los diversos curados realizados sobre el dataset, éstos tienen al menos un experimento exitoso de acoplamiento con un ligando de la tabla `reduced_ligands.csv`.

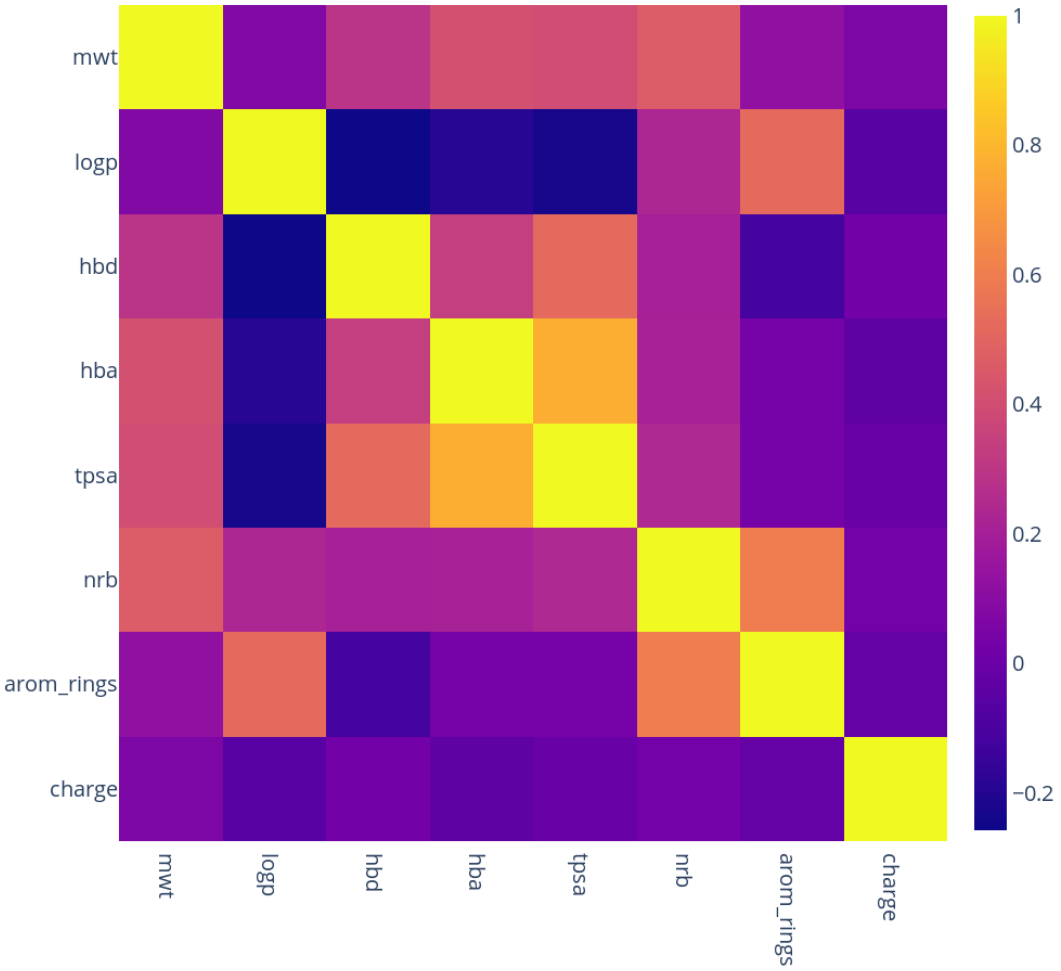


Figure 2.15: Heatmap de la correlación de Spearman entre features de ligandos del archivo `ligands.csv`.

2.2.5 Análisis de Componentes Principales

Otro aspecto interesante a estudiar es qué información otorga cada propiedad a la hora de explicar los datos. Para esto decidimos usar una técnica denominada PCA (del inglés, *Principal Component Analysis*), la cual construye lo que denominamos *componentes* como combinaciones

lineales de las propiedes originales, guiándose por el orden de la variabilidad total que éstas presentan en la muestra. Esto puede verse explicado para un caso bidimensional en la figura 2.16, en donde los puntos azules representan las muestras y la flecha verde representa el vector de una de las componentes calculadas por PCA. Mientras más se dispersen los datos en alguna dirección, más relevante será la componente. Por otro lado, la técnica tiene como requisito que no exista una marcada correlación entre las variables, lo cual podemos evaluar teniendo en cuenta los análisis de correlación anteriormente hechos.

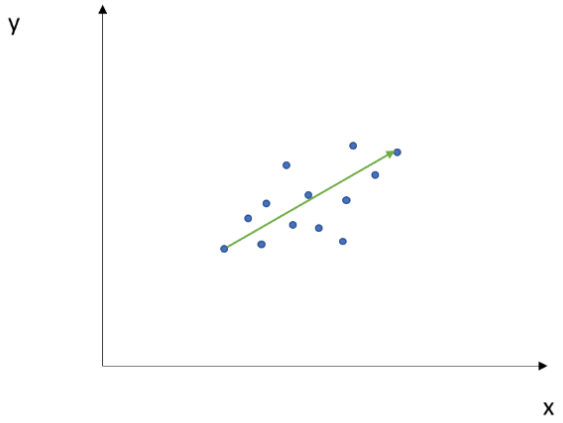


Figure 2.16: PCA aplicado a un conjunto de muestras (puntos azules).

Como mencionamos anteriormente, la bibliografía[54] sugiere varias maneras de interpretar los resultados de Spearman en función de distintos umbrales. Nosotros tomamos una versión más bien conservadora, y consideraremos que existe una correlación fuerte únicamente para valores de los coeficientes de entre 0.8 y 1.
Por ello, para pockets, eliminamos (de forma arbitraria) las propiedades:

- `sasa_total`, dado que presenta una fuerte correlación tanto con `sasa_polar`, `sasa_apolar` y `volume`.
- `apolar_alpha_sphere_proportion`, que presenta una fuerte correlación con `mean_local_hydrophobic_density` y `proportion_polar_atoms`.

Por el lado de las propiedades de los ligandos (figura 2.15), no parecen presentar correlación fuerte entre ninguna de sus propiedades, de forma que no realizamos ninguna modificación[j].
Podemos usar PCA de formas muy útiles. Tomando estas nuevas componentes, podríamos optar por quedarnos únicamente con las de mayor valor y plantear usarlas como nuevas *features* para nuestro estudio, descartando las originales, algo que se conoce como *feature extraction*. Otro caso interesante es el de evaluar el peso que tiene cada una de las features originales sobre estas nuevas componentes, lo cual nos da a su vez una idea de la importancia que tienen dichas features a la hora de explicar la variabilidad de los datos. En base a esto podemos elegir focalizarnos en las features que más valor aportan a las componentes mejor rankeadas, algo conocido como *feature selection*[55].
Con respecto a las opciones mencionadas, optamos en este punto por un enfoque más bien conservador: no usar PCA como *feature extraction* para no descartar ninguna propiedad. El motivo es que este es un trabajo interdisciplinario y tomar esta clase decisiones requiere de un conocimiento muy específico de los datos y del dominio, lo cual excedía este trabajo de tesis. En particular, tampoco usamos PCA como *feature selection* para quedarse con un subconjunto de features, pero sí nos valimos del análisis que hace para entender qué features tienen más relevancia. Usando la implementación de `scikit-learn` aplicamos PCA a los datasets de pockets

[j] Los motivos por los cuales la técnica puede presentar problemas se detallan en el Apéndice A, secciónA.2.10.

y de ligandos, previamente estandarizados en todos sus valores. La estandarización es importante dado que si existen valores muy extremos de las propiedades para algunas muestras, esto puede desbalancear bastante los puntajes resultantes de PCA para cada una de ellas, no permitiendo tomar estos resultados con confianza para un posterior análisis.

2.2.5.1 Pockets

Se graficó el porcentaje de varianza que explica cada componente, usando un *scree plot* (un tipo de gráfico que muestra el valor correspondiente a cada componente principal en orden decreciente). Como se puede ver en la figura 2.17, las primeras cinco componentes principales, en conjunto, explican el 90% de la varianza.

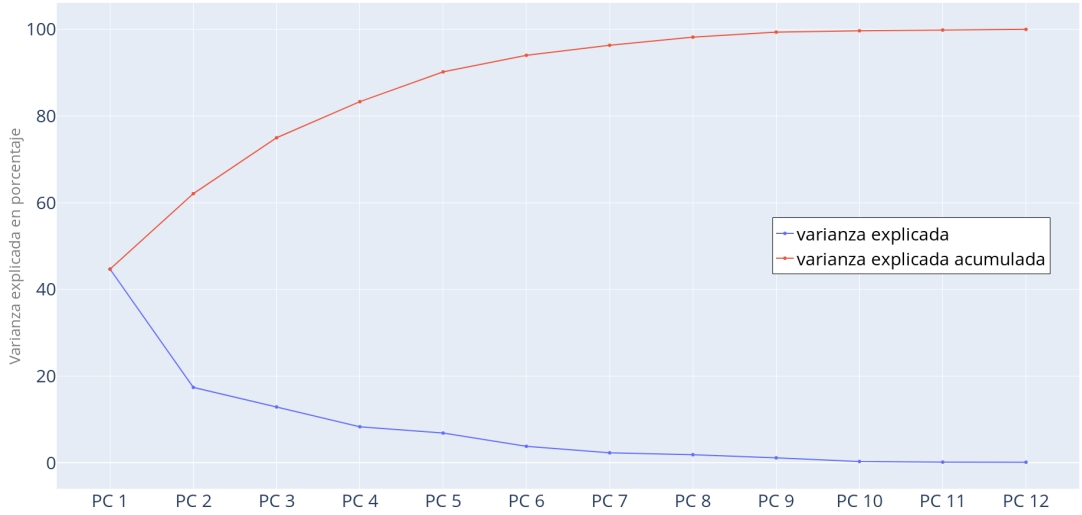


Figure 2.17: Varianza explicada y varianza explicada acumulada para componentes principales en base a propiedades de bolsillos. Puede verse claramente que las primeras 5 componentes describen más del 90% de la varianza total.

Adicionalmente, para visualizar los coeficientes de cada variable original dentro de las componentes, se utilizó un *heatmap*, el cual se puede observar en la figura 2.18.

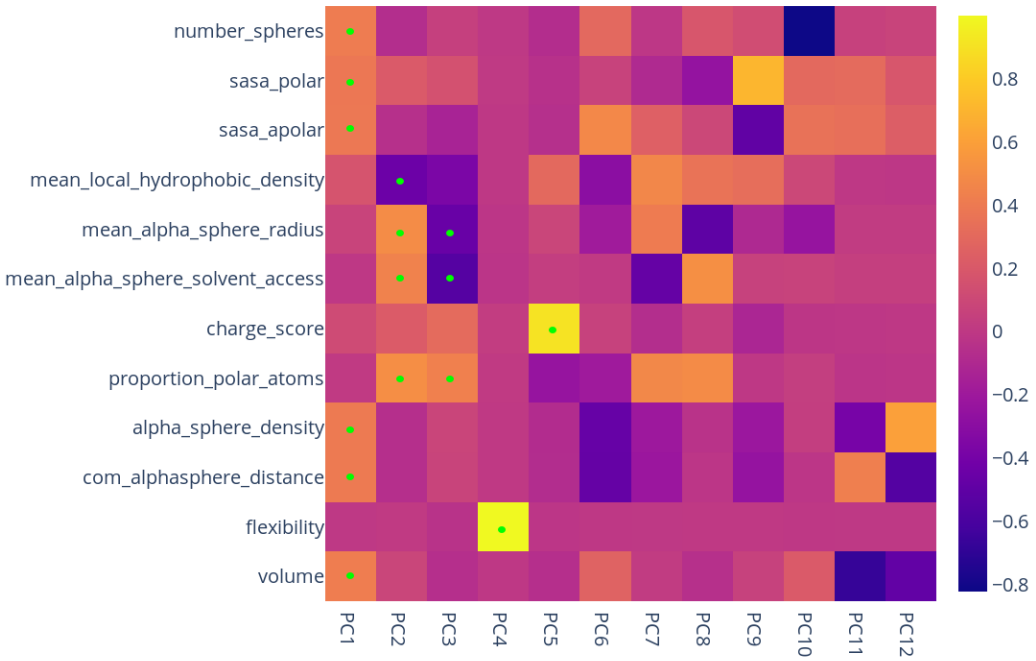


Figure 2.18: Heatmap correspondiente a las PCA de pockets. El puntaje representa los autovalores de los vectores que definen la distribución de la componente en la dimensión de las propiedades. Los puntos verdes indican los valores más altos en módulo para cada componente.

En la figura 2.18 se ve que, en la primera componente, tienen un fuerte peso las features `number_spheres`, `sasa_polar`, `sasa_apolar`, `alpha_sphere_density`, `com_aplha_distance` y `volume`. Todas ellas se encuentran relacionadas al área o volumen que describe el pocket. Por otro lado, la segunda y tercera componente se encuentran en gran parte definidas por las propiedades físico-químicas como la polaridad en `proportion_polar_atoms` o hidrofobicidad en `mean_local_hidrophobic_density` y, por otro, propiedades más bien relacionadas a las alpha-spheres producidas por fpocket, como `mean_alpha_sphere_radius` y `alpha_sphere_solvent_access`. En cuanto a la cuarta y quinta componente, se encuentran prácticamente definidas por `flexibility` y `charge_score` respectivamente.
Esta información reorientó nuestro trabajo, porque dada la importancia que el volumen del bolsillo reportaba en la variabilidad general, intuímos que dicha feature podía ser muy relevante en el poder predictivo de nuestros futuros modelos. Una eventual disección del volumen en componentes que explicaran su geometría tomó una relevancia inusitada en el desarrollo posterior de esta tesis (ampliaremos más adelante).

2.2.5.2 Ligandos

De manera análoga al estudio realizado sobre los pockets, corrimos PCA sobre los ligandos, cuyos resultados pueden verse en las figuras 2.19 y 2.20.

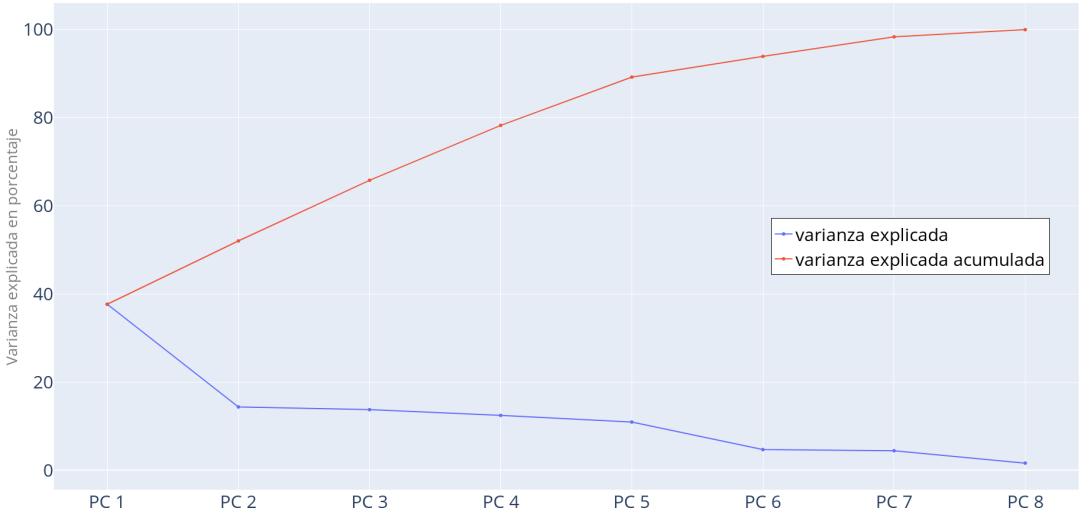


Figure 2.19: Varianza explicada y varianza explicada acumulada para componentes principales en base a propiedades de ligandos. Puede verse claramente que las primeras 5 componentes describen más del 90% de la varianza total.

Como se puede observar, si bien las curvas de la figura 2.19 (ligandos) se asemejan a las obtenidas en la figura 2.17 (pockets), en el caso de los ligandos el crecimiento de la varianza explicada acumulada para las primeras componentes es lineal. Más allá de esta diferencia, nuevamente, con las primeras cinco componente se llega a explicar el 90% de la varianza.

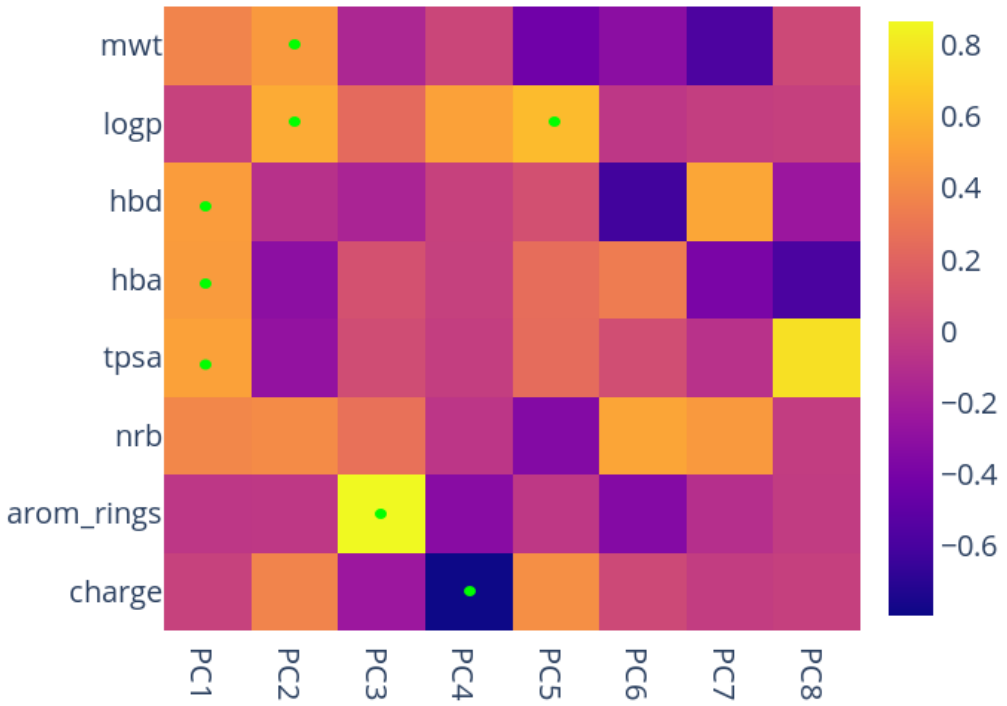


Figure 2.20: Heatmap correspondiente a las PCA de ligandos. El puntaje representa los autovalores de los vectores que definen la distribución de la componente en la dimensión de las propiedades. Los puntos verdes indican los valores más altos en módulo para cada componente.

De la figura 2.20 se puede concluir que la primera componente se encuentra principalmente influenciada por la feature `tpsa` (muy relacionada con el concepto de polaridad). En menor medida, también se ve influenciada por las propiedades `hba` y `hbd`. En el caso de la segunda componente, la misma puede estar asociada a `logp` y `mwt`, propiedades sin relación aparente entre sí. La tercera componente se relaciona con la propiedad `arom_rings`, mientras que en la cuarta componente del PCA predomina `charge`. Finalmente, en la quinta componente, aparece nuevamente `logp` como propiedad predominante.

2.2.6 Aproximación de la forma de pockets y ligandos

Considerando que el análisis de la sección anterior evidenció el gran protagonismo que tiene la geometría del pocket para explicar la dispersión total de los datos, decidimos incorporar al dataset propiedades que describen dicha característica. A su vez, dado que se espera una correspondencia entre geometrías de pockets y ligandos, se decidió incluir la forma de estos últimos también. Para obtener una medida del volumen elegimos una aproximación mediante un elipsoide[56], una figura geométrica tridimensional suficientemente versátil en este contexto. Teniendo en cuenta que un elipsoide está definido por sus 3 radios, exploramos distintas alternativas para poder calcularlos. Entre ellas, podemos mencionar métodos basados en el cálculo del momento de inercia principal o PMI[57](una medida de la inercia rotacional de un cuerpo), métodos basados

en PCA[58] (técnica detallada en la sección anterior) o bien enfoques basados en el cálculo del elipsoide de volumen mínimo que recubre un conjunto dado de puntos (en inglés, *Minimum Volume Enclosing Ellipsoid*[59], abreviado MVEE).
En una primera instancia exploramos métodos basados en el momento de inercia. Encontramos que, si bien es perfectamente factible su utilización en nuestro caso, deben llevarse a cabo ciertos ajustes para garantizar la precisión de sus resultados. En particular, al ser el momento de inercia una medida física, la masa juega un rol muy importante dentro de la medición, dado que éste "refleja la distribución de masa de un cuerpo respecto a un eje de giro". Suponiendo entonces que contamos con dos moléculas de tamaño similar pero con masa notoriamente distinta (por ejemplo, debido a su conformación atómica), inevitablemente esto redundará en momentos de inercia a su vez distintos. Una posible alternativa en este punto sería "remover" la masa de la ecuación, por ejemplo asumiento una masa uniforme para todos los átomos de la molécula. Sin embargo, nuevamente podríamos encontrar inconvenientes en este caso, a causa de regiones con una densidad atómica pronunciada en su estructura. En dicho caso, la masa acumulada jugará un rol importante en los cálculos de PMI. En ambos casos, será entonces necesario recurrir a una etapa de post-procesamiento, en donde se realice una suerte de "escalado" de los valores, a fin de preservar la correspondencia entre los distintos pockets y ligandos de nuestro dataset, y así poder llegar a plantear una idea de complementariedad geométrica o *encastre*. Estos problemas que nos planteamos, son los que nos llevaron a descartar usar momentos de inercia para caracterizar la conformación espacial de un pocket o ligando.[k]
En el caso de PCA, es perfectamente factible su utilización, pero dado que es una medida de análisis de dispersión de los datos, corremos el riesgo de que si los átomos del pocket se encuentran dispuestos de maneras muy extremas (por ejemplo muchos átomos juntos y uno lejano), nos encontraremos que las medidas que PCA nos otorga no reflejarán realmente las dimensiones del pocket o del ligando. Por esta razón decidimos descartar PCA y optar por la técnica de MVEE que se detalla a continuación.
A diferencia de los métodos anteriores, el método de MVEE aparece como un método más simple. Se basa en cálculos puramente geométricos: partiendo de un conjunto de puntos en el espacio, construye el elipsoide de mínimo volumen que envuelve a todos ellos. Es principalmente por esta razón que decidimos adoptarlo para el presente trabajo, en donde los puntos en el espacio pueden ser o bien los átomos de la molécula para los ligandos, o bien las α-spheres para el caso de los pockets. La razón por la cual decidimos utilizar las α-spheres de los pockets y no los átomos que lo componen se encuentra detallada en las secciones siguientes.
Para la implementación del algoritmo, utilizamos un proyecto público de github[60] basado en el método de Khachiyan[61], el cual aproxima los valores de modo iterativo y nos permite obtener los tres radios principales del elipsoide, ordenados de mayor a menor. En la figura 2.21 detallamos un caso de ejemplo para el cálculo de MVEE. A la izquierda se puede ver una lista con las coordenadas en el espacio de los centros de las α-spheres para un pocket de la proteína con Uniprot ID `O60885`. Debajo se detallan los resultados del cálculo de MVEE, en particular los 3 radios del elipsoide construído. A la derecha, se puede observar una una visualización de dicho elipsoide en el espacio.

[k] Para más detalle acerca de estas problemáticas, instamos al lector a consultar el Apéndice A, sección A.2.11.

```
# O60885 pdb entry
# Centers of alpha-spheres for pocket 0
# (file pocket0_vert.pqr)
pts = np.array([
  [23.325, 47.366, 1.236],
  [23.115, 47.198, 1.472],
  ...
  [22.516, 47.394, 1.408],
  [19.892, 47.898, 0.960],
  [22.136, 46.955, 0.742]
])

# Results
volume: 330.301570788s
radii: [ 2.37 4.17 7.98]
center: [ 22.91 48.07 5.73]
rotation: [[-0.64 -0.67 0.37]
          [ 0.76 -0.58 0.29]
          [ 0.02  0.47 0.88]]
```

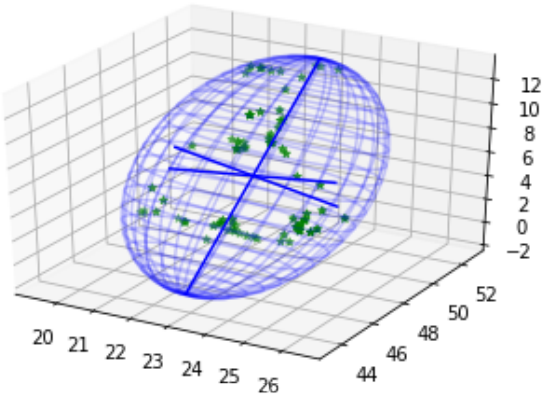


Figure 2.21: Minimum Volume Enclosing Ellipsoid (MVEE) para el pocket número 0 de la proteína con Uniprot ID`O60885`. Los ejes se corresponden con las coordenadas cartersianas. Los valores se encuentran en Angstroms (10^{-8} cm).

2.2.6.1 Cálculo de la geometría de los pockets

Computamos el MVEE de cada bolsillo a partir del output generado por fpocket para cada uno de ellos, el cual fue provistos por el BIA. De cada uno de ellos se tomó la información referente a las α-spheres detallada en los archivos `*_vert.pqr`, en un extenso procesamiento que se encuentra descripto en el Apéndice A, sección A.2.11.1. La razón por la que optamos por usar las α-spheres y no los átomos que definen al bolsillo (detallados en los archivos `*.pdb`) tiene que ver con que el MVEE arrojaría un volumen mayor al que realmente existe, mientras que si utilizamos las α-spheres como si fueran átomos de un molécula ficticia, su MVEE se aproximará mejor al valor esperado.
Durante la generación de MVEE para los pockests hubo errores en algunos casos particulares[l]. Al no contar con otra forma de obtener estos valores, y considerando que no representaban un porcentaje significativo, decidimos eliminar los registros correspondientes de los archivos `pockets.csv` y `bindings.csv`. Finalmente, una vez generados los elipsoides para cada pocket, calculamos los valores de sus tres radios y los adjuntamos al conjunto de propiedades de pockets definidas en el archivo CSV correspondiente bajo lo siguientes nombres: `lr` (large radius), `mr` (medium radius) y `sr` (small radius).

2.2.6.2 Cálculo de la geometría de los ligandos

Al igual que en el caso de los pockets, calculamos el MVEE para cada uno de los ligandos. Para ello, en primer lugar, obtuvimos los archivos `sdf` correspondientes de RCSB[m], en donde se define la estructura bidimensional de la molécula. A partir de esta información, calculamos el modelo 3D correspondiente mediante técnicas que se encuentran detallados en el Apéndice A, sección A.2.11.3. Finalmente, aplicamos el método MVEE sobre la estructura 3D y medimos sus tres radios para adjuntarlos, ordenados de menor a mayor, a la información de los ligandos en el archivo `csv` correspondiente.

[l] Para ver más detalle de esto, consultar el Apéndice A, sección A.2.11.2.
[m] Para más detalle de este proceso, consultar el Apéndice A, Sección A.2.11.3.

2.2.6.3 Distribución de los valores de MVEE

A continuación detallamos las distribuciones de los valores de los 3 radios del elipsoide (`lr`, `mr` y `sr`) para pockets, ligandos y ligandos con interacciones, respectivamente.

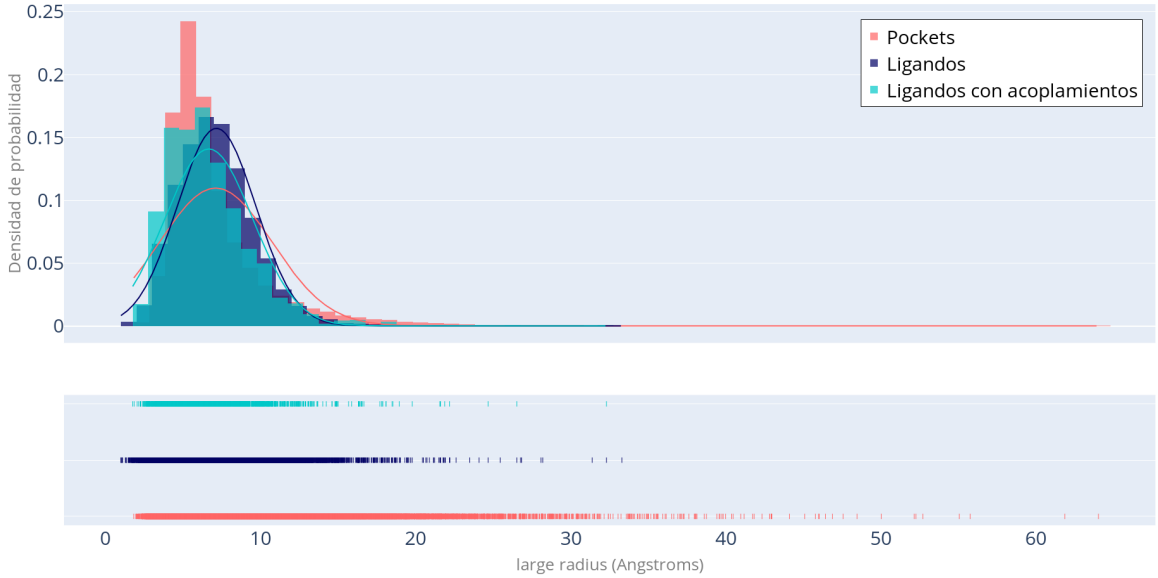


Figure 2.22: Distribución de valores de `large radius`.

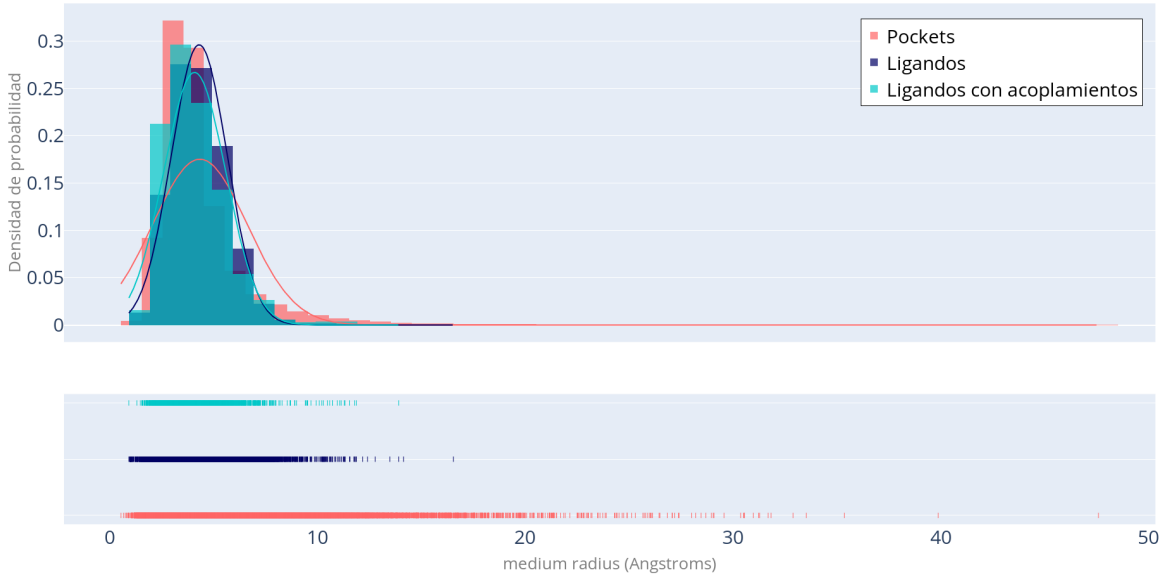


Figure 2.23: Distribución de valores de `medium radius`.

Figure 2.24: Distribución de valores de `small radius`.

En las tres figuras anteriores (figuras 2.22, 2.23 y 2.24) puede notarse que la distribución de las frecuencias para cada caso es similar tanto para ligandos (con o sin acoplamientos) y pockets. En el caso de los pockets en particular, mirando la distribución de los valores en la sección inferior de los tres gráficos, puede notarse que poseen más valores elevados que los ligandos. Esto es esperable dado que los ligandos suelen ser moléculas muy pequeñas, mientras que los pockets dependen de la estructura proteica, la cual puede ser muy variable.
Por otro lado, los ligandos con interacciones parecen tener valores ligeramente más pequeños que el total de los ligandos presentes en `ligands.csv`, indicando que las moléculas más bien grandes no suelen presentar tantos acoplamientos como las más pequeñas.
Por último, decidimos chequear si no existe algún tipo de correspondencia entre estas nuevas propiedades de bolsillos y ligandos, a fin de evaluar si es posible predecir unas en función de las otras. Para ello construimos un nuevo *heatmap* en función de la matriz de correlación de Spearman, que puede verse en la figura 2.25. Es claro que los coeficientes de Spearman entre propiedades de uno y de otro son muy bajos, prácticamente nulos. Las correlaciones entre `lr`, `mr` y `sr` entre pockets y entre ligandos tiene algunos valores más altos, sin embargo ninguno llega al umbral de 0.8 elegido para decir que existe una correlación fuerte.

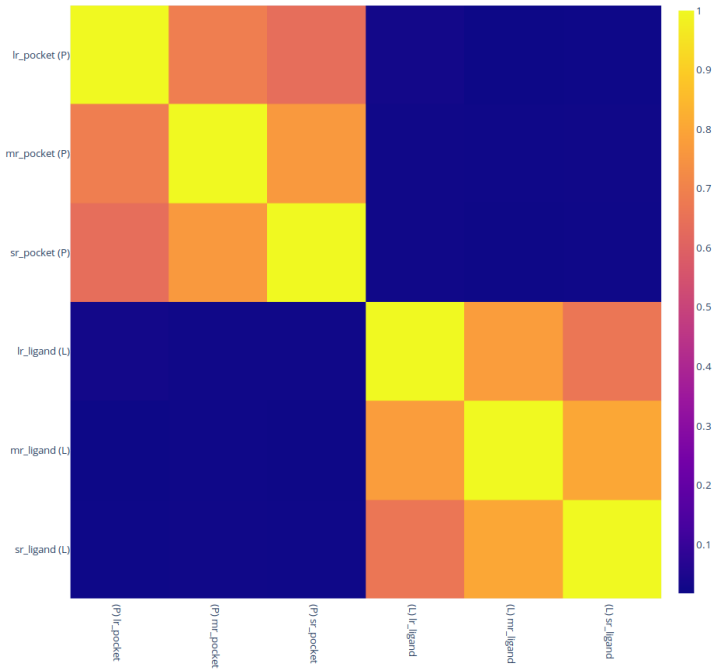


Figure 2.25: Heatmap del coeficiente de correlación de Spearman correspondiente a las features `lr`, `mr` y `sr` de `pockets.csv` y `ligands.csv`, asociadas a través de `bindings.csv` Las propiedades que figuran con `(P)` son las de pockets y con `(L)` las de ligandos.

Estos últimos resultados pueden indicar que la descripción de la forma geométrica de los bolsillos/ligandos a través de radios de un elipsoide quizás no sea la mejor alternativa. Pueden existir regiones con protuberancias o regiones más bien vacías que cumplen un rol importante en el acoplamiento, y la manera que elegimos de modelar el volumen no logra capturar la complejidad de esas formas. También puede ocurrir que la geometría de bolsillos y ligandos mute en el tiempo, por ejemplo en función de factores externos como los solventes en los cuales se da el acoplamiento. De todas formas, al momento no tenemos herramientas suficientes para considerar que estos descriptores no puedan ser de utilidad para un sistema de recomendación, de manera que decidimos conservarlos.

2.2.7 Extensión del dataset usando datos de ChEMBL

El análisis de clústers con proyección de labels (detallado en la sección 2.2.2) para el caso de ligandos → pockets no presentó resultados tan buenos como los de pockets → ligandos. Como mencionamos anteriormente, nuestra hipótesis frente a la presencia de este fenómeno es que se debe a la poca cantidad de ligandos con los que contamos. Para aumentar la cantidad de registros, se propuso tomar información de una nueva fuente de datos: ChEMBL[n]. La principal razón por la cual no se incorporaron datos de esta fuente desde un primer momento, es que los experimentos que allí se detallan describen ensayos en donde se busca consignar la interacción entre un ligando y una proteína, pero no se especifica en qué pocket de la proteína sucede dicho acoplamiento. Es decir, no hay estudios cristalográficos de donde se pueda obtener esta infor-

[n] Para ver el detalle de esta base de datos consultar el Apéndice A, sección A.2.1.

mación. Teniendo esto último presente, el BIA confeccionó un nuevo dataset utilizando datos obtenidos de ChEMBL. Para esta nueva etapa, trabajamos bajo la hipótesis de que, de todos los bolsillos disponibles de la proteína, el acoplamiento del ligando se produce con el *mejor* de ellos: el sitio activo. La elección de este bolsillo es sencilla considerando que `fpocket` computa una medida de scoring a estos fines, denominada `druggability score`. Mientras más alto sea este valor para un pocket, más "drogable" es el pocket en sí y, por lo tanto, `fpocket` lo ubica más alto en el ranking de pockets calculados. De esta forma, el pocket número 0 se presenta como el más adecuado para un eventual acoplamiento, y, por lo tanto, es el que se eligió para todos los casos.
Con respecto a los ligandos, los datos de ChEMBL se actualizan de forma continua[o]. Esto lo pudimos corroborar haciendo un estudio exploratorio. Encontramos que para algunos ligandos la información aportada por el BIA difería con respecto a la que figura en la página web de ChEBML. En particular, al menos la mitad de los ligandos del dataset inicial tenía valor 0 en la propiedad `tPSA`, que no es lo que la bibliografía sugiere respecto a la distribución de esta variable[62]. Es por eso que descargamos la última versión de ChEMBL(chembl[63], 2019) y actualizamos los datos que teníamos (para más detalle, consultar el Apéndice A, sección A.2.12).
En este proceso, no pudimos conseguir la información correspondiente a la propiedades `charge`. La misma no se encuentra definida como descriptor en la tabla `public.compound_properties` de ChEMBL. Antes de encarar un nuevo desarrollo para calcular este valor, analizamos los gráficos de PCA sobre el dataset original y notamos la poca relevancia que tenía dicha propiedad, por lo que decidimos eliminarla del dataset.
Por último, realizamos comparaciones de los histogramas de todos los descriptores para compararlos con los de la versión inicial del dataset. Los resultados tanto para propiedades de pockets como para de ligandos fueron muy similares[p].

Análisis de interacciones proyectadas

Realizamos un nuevo análisis de interacciones proyectadas de ligandos a pockets, cuyos gráficos pueden observarse en la figuras 2.26 y 2.27.

[o] Toda esta información se encuentra descripta en el FAQ del sitio web oficial de ChEMBL: `https://chembl.gitbook.io/chembl-interface-documentation/frequently-asked-questions`. Algunas breves precisiones que podemos darle de forma rápida al lector: *The data is updated regularly, with releases approximately every 3-4 months.*

[p] Este material, al igual que todas los gráficos que figuran en este trabajo, pueden encontrarse listados en el Apéndice C (Material suplementario).

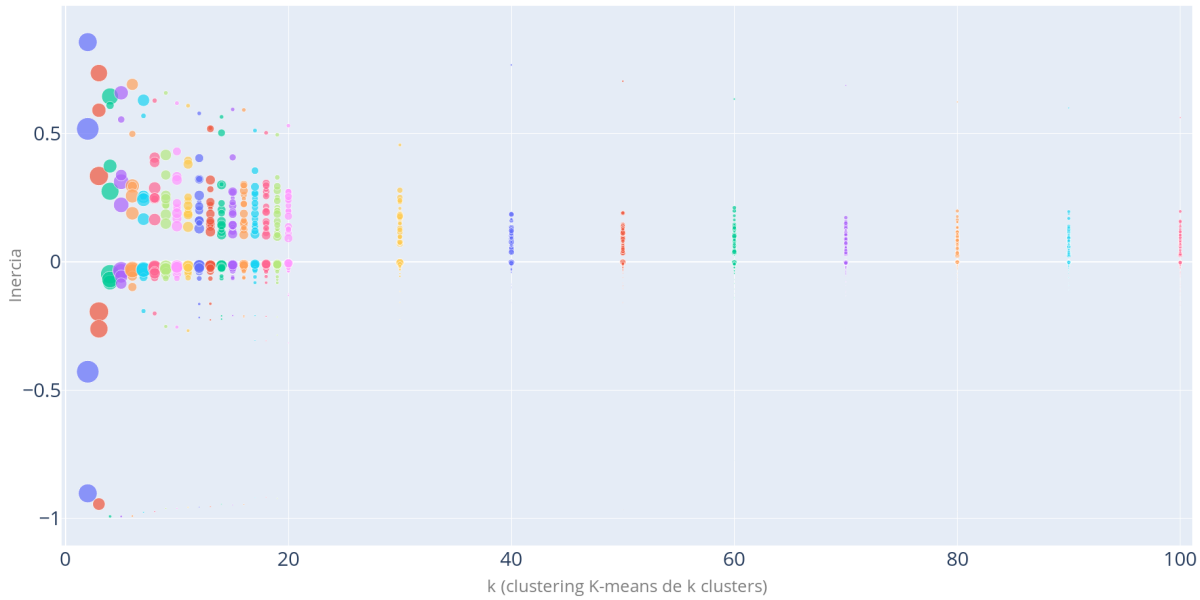


Figure 2.26: Bubblechart de clustering de pockets variando el número de clusters k usado como parámetro de K-means de 2 a 100. En la mitad inferior se visualizan los clusterings de pockets y en la parte superior los de ligandos.

Al comparar la figura 2.26 con la figura 2.3, notamos que hubo una mejora en el clustering de ligandos (parte inferior). Los clusters con mayor cantidad de elementos cuentan con una inercia que, a medida que aumenta el valor de k, rápidamente se acerca a 0.

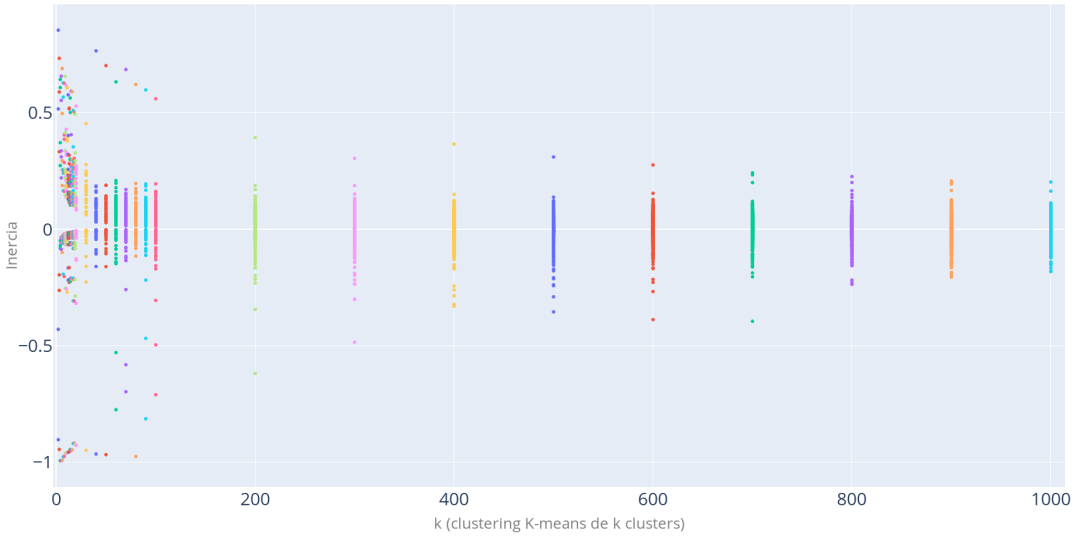


Figure 2.27: Scatterplot variando el número de clusters k de K-means de 2 a 1.000. Cada punto representa un cluster. En la mitad inferior se visualizan los clusterings de pockets y en la mitad superior los de sus ligandos asociados.

En la figura 2.27, si la comparamos con la figura 2.4 la mejora en líneas generales es más bien leve pero marcada. A medida que aumenta el valor de k el clustering, los pockets proyectados

muestran una tendencia y una inercia que va siempre reduciendo su valor. A diferencia de lo que ocurría en la figura 2.4, donde la inercia de los clusters extremos parecía oscilar a medida que el valor de k aumentaba, en este nuevo gráfico parece converger más notoriamente. En la comparación, con respecto a los valores absolutos de inercia, notamos una disminución.

2.2.7.1 Cálculo de la geometría de pockets y ligandos de ChEMBL

De manera análoga a lo realizado para el dataset inicial, computamos los elipsoides para los pockets y ligandos provenientes de ChEMBL.
Para el caso de los pockets, dado que la información de las corridas de `fpocket` fue brindada por el BIA en el mismo formato, el procesamiento fue prácticamente el mismo que el detallado en la sección 2.2.6.1[q].
Para el caso de los ligandos, ChEMBL provee la información de la estructura 2D de los compuestos, la cual utilizamos para calcular la estructura 3D de mínima energía mediante un proceso ya mencionado anteriormente (detallado en el Apéndice A, sección A.2.11.3). Los datos figuran en las entradas de la base de datos `sql` para cada compuesto, y a su vez se encuentran disponibles mediante un único archivo `sdf` con las estructuras de todos los compuestos existentes en la base de datos. En particular, decidimos trabajar con la última alternativa para acelerar los cálculos, utilizando el módulo `split_sdf.py` para dividir el SDF en partes. Dichas partes pueden usarse como entrada para correr en paralelo el script de cálculo de MVEE[r]. A su vez, el script ha sido programado teniendo en cuenta posibles cortes en el procesamiento, ya sea por cuestiones de errores en el archivo o errores de cantidad de memoria utilizada o fallas de hardware o suministro eléctrico, de forma que, puede relanzarse y restaurar el punto en el cual ocurrió la interrupción. Los tres radios generados fueron adjuntados a cada entrada del archivo `ligands.csv`, respetando el orden de menor a mayor.

2.2.8 Estructura final del dataset

Finalmente, el dataset final quedó conformado por los siguientes archivos:

- `unified_pockets.csv`: pockets con sus propiedades, identificados por su `unified_pocket_id` (71.826 registros).
- `ligands.csv`: compuestos con sus propiedades (166.189 registros).
- `bindings.csv`: interacciones unified_pockets y ligandos de los dos archivos mencionados previamente (239.878 registros).
- `reduced_ligands.csv`: subconjunto de compuestos con sus propiedades. Incluye sólo a aquellos compuestos de `ligands.csv` que participan en interacciones (archivo `bindings.csv`) (84.277 registros).

Adicionalmente conservamos:

- `annotated_pockets.csv`: correspondencia entre identificadores de `unified_pockets.csv` y del archivo `pockets.csv` original (misma cantidad de registros, 71.826).

[q] Detallado en el Apéndice A, sección A.2.11.1.

[r] Los archivos usados para este procesamiento se encuentran listados en el Apéndice C (Material Suplementario).

Las propiedades de `unified_pockets.csv` y `reduced_ligands.csv` que utilizamos en la etapa de aprendizaje automático son 16 features para pockets y 10 features para ligandos y se listan en la Table 2.7.

pockets	**ligandos**
number_spheres	mwt
sasa_total	logp
sasa_polar	hbd
sasa_apolar	hba
mean_local_hydrophobic_density	tpsa
mean_alpha_sphere_radius	nrb
mean_alpha_sphere_solvent_access	arom_rings
apolar_alpha_sphere_proportion	lr
charge_score	mr
proportion_polar_atoms	sr
alpha_sphere_density	
com_alphasphere_distance	
flexibility	
lr	
mr	
sr	

Table 2.7: Features utilizadas en la etapa de aprendizaje automático para caracterizar a los pockets y ligandos, respectivamente.

2.2.8.1 Sobre-representación de pockets en acoplamientos: solución propuesta

Durante el análisis de la distribución de acoplamientos por pocket de ChEMBL, encontramos que alrededor del 1% de éstos participaba en dos tercios de la cantidad total de interacciones de nuestro dataset. Cada uno de estos pockets se unía con por lo menos 10 ligandos, llegando en casos excepcionales a más de 20.000. Conjeturamos que estas características se deben a la presencia de un sesgo experimental en los datos sobre interacciones provenientes de ChEMBL. Con el fin de intentar reducir el impacto de este desbalance en los datos, decidimos generar un dataset alternativo, en el cual conservamos sólo aquellos pockets con a lo sumo 8 acoplamientos. Este valor fue elegido en base a la cantidad máxima encontrada en el dataset de PDB (ver figura 2.5), como forma de mantener la consistencia tras la incorporación de los datos de ChEMBL. Los archivos conservan los mismos nombres que los mencionados antes. El único cambio fue la cantidad de registros en cada uno de ellos (con excepción de `ligands.csv`), que se vió reducida en menor o mayor medida, indicada a continuación:

- `reduced_ligands.csv`: 2.597 registros.
- `pockets.csv`: 71.651 registros.
- `annotated_pockets.csv`: 71.651 registros.
- `bindings.csv`: 80.259

Como podemos observar, la cantidad de ligandos que participan en interacciones se redujo considerablemente: un 3% de la cantidad total previa al filtrado. Esto representa un retroceso

con respecto a la motivación de incorporar datos de ChEMBL, que era poder contar con mayor cantidad de muestras para entrenar los modelos de aprendizaje automático. Por esta razón, tomamos la decisión de conservar tanto el conjunto sesgado como el conjunto no-sesgado (que posee un máximo de 8 acoplamientos diferentes por pocket) y evaluar el desempeño de los algoritmos de aprendizaje con ambos. Nos referiremos al primer conjunto como "dataset sesgado", y al segundo conjunto como "dataset 8L" (en referencia al límite de cantidad de acoplamientos mencionado). Estos conjuntos pueden observarse en la figura 2.28.

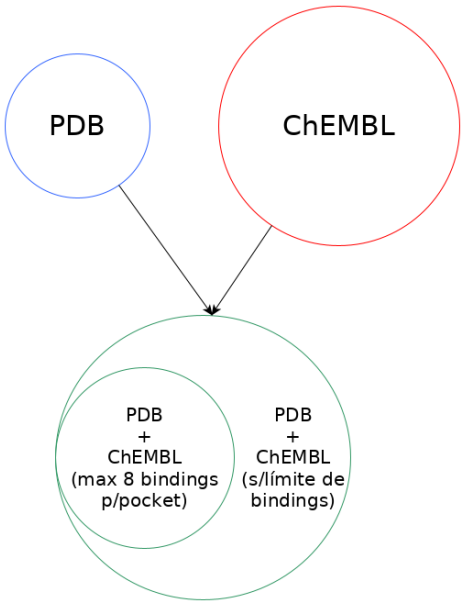


Figure 2.28: A partir de los datasets de PDB (azul) y ChEMBL (rojo), formamos un nuevo conjunto de datos (verde), del cual vamos a tomar 2 variantes: *dataset 8L* y *dataset sesgado* (sin límite de bindings).

2.3 Conclusiones

Gran parte del tiempo que dedicamos a esta etapa de construcción del dataset rondó en identificar qué información de pockets/ligandos existía en bases de datos de público acceso, y determinar si era lo suficientemente buena como para encarar las etapas posteriores de este trabajo de tesis. Lamentablemente no logramos hallar suficiente bibliografía que hiciese un análsis crítico de los descriptores de pockets proporcionadas por `fpocket` ni de las características de ligandos detalladas en ChEMBL, ZINC y PDB. Tampoco encontramos argumentos para optar por incluir más descriptores de los que se habían seleccionado originalmente para el desarrollo de LigQ. En particular, la recomendación por parte del BIA de no tomar propiedades de scoring, debido a que las mismas se computan a partir de otras propiedades ya presentes en el dataset, nos llevó también a descartar cualquier propiedad calculable a través de, por ejemplo, bibliotecas como JChem [64]. Todas estas cuestiones nos llevaron a plantearnos si la cantidad de propiedades consdieradas era suficiente para que los resultados al aplicar nuestras técnicas de Machine Learning fuesen satisfactorios. Debido a que nuestro conocimiento del área de estudio es limitado, tomaron cierta relevancia cuestiones más bien numéricas de los descriptores, analizando distribuciones de los datos y probando diversas herramientas para poder inferir la forma que toman. Teniendo en

cuenta los plazos de este trabajo, la complejidad de las bases de datos de las cuales extrajimos la información[s] y , nuevamente, nuestro conocimiento del área, creemos que esta decisión fue acertada.
A su vez, el éxito de los resultados a través del uso de técnicas de Machine Learning no sólo depende de la cantidad de propiedades utilizadas sino que también se puede ver afectado por diversos factores, como por ejemplo, la cantidad y calidad de datos con que se cuenta. En consonancia con esta reflexión, otro objetivo tuvo que ver con comprender las razones de elegir un determinado diseño del dataset por sobre otro. Esto provocó que, muchas de las ideas fueran surgiendo sobre la marcha, reconfigurando las hipótesis que nos habíamos planteado inicialmente. Así, tuvimos que restructurar múltiples veces el dataset hasta llegar a la versión final que, por cierto, dista de ser nuestra ideal. En suma, la etapa de análisis exploratorio de datos nos llevó gran parte del desarrollo total del trabajo y quedaron muchas alternativas por probar, las cuales describimos con detalle en el capítulo 8 (Trabajo Futuro).
Con respecto a este último punto, podemos mencionar que primó también un objetivo secundario: el deseo de poder dejar un precedente conciso y detallado a las áreas involucradas que puedan utilizar esta información para trabajos futuros, respetando el precepto de reproducibilidad de los resultados conseguidos, a partir de los materiales y métodos descriptos, como así también el detalle de todos los inconvenientes y ventajas de cada uno. Es por eso que, además, quedan a disposición tanto los *scripts* como las jupyter notebooks utilizadas, las cuales cuentan con documentación autocontenida que explica en detalle cada una de las etapas de los procesos y permite generar los resultados de forma prácticamente automática.

[s] Citado del FAQ de ChEMBL:

> Why are there so many different types of Standard Units in the database?
> The published units are taken directly from the literature and we then attempt to standardise these to report as a standard type. This is an ongoing curation task, which was started with the most common units first."

3 | Predicción proteína-ligando

> Machine learning depends heavily on data. It's the most crucial aspect that makes algorithm training possible and explains why it became so popular in recent years. But regardless of your actual terabytes of information and data science expertise, if you can't make sense of data records, this technique will be nearly useless or perhaps even harmful. [65]

En este capítulo detallaremos los métodos utilizados para construir los modelos de predicción que estimen la probabilidad de acoplamiento de nuevos compuestos a sitios conocidos de una proteína (drug-discovery). Finalmente, discutiremos los resultados obtenidos.

3.1 Técnicas evaluadas y pipeline propuesto

Un sistema de recomendación es un software que proporciona sugerencias de utilidad para un usuario, en base al estudio de sus datos personales, preferencias y las características de los ítems de interés. Existen distintos tipos de sistema de recomendación según la técnica computacional que se utilice[35]:

- filtro de contenido o *item based*: a partir del conocimiento que se tiene de los items que el usuario ha valorado, se le recomendarán aquellos que sean similares a los que le hayan resultado interesantes.
- filtro demográficos o *user based*: se realizan en función de las características de los usuarios (edad, sexo, situación geográfica, profesión, etc). Se busca recomendar items que sean de interés para usuarios similiares.
- filtro colaborativo basado en memoria: similar al caso anterior, pero la medida de similitud no está relacionada a datos demográficos, sino a los items que resultaron de interés para cada uno. Es decir, "si a tal usuario le gustaron las mismas películas que a mí, hay altas chances de que si le gusta un nuevo estreno, a mí también me guste". Para ello se calculan los items que han sido votados por ambos usuarios y se comparan dichos votos para calcular su parecido o cercanía.
- filtro colaborativo basado en modelos: utiliza modelos estadísticos. Aquí aparece el concepto de **aprendizaje automático**.

Para trazar un paralelismo con estos ejemplos, en un problema de *Virtual Screening* podemos pensar que las proteínas representan a los "usuarios" y los ligandos representan a los "items". Para el caso de *Inverse Virtual Screening* se plantea el mismo paralelismo pero invertido: los ligandos representan a los "usuarios" y las proteínas representan a los "items".
En este sentido, tanto los filtros basados en contenidos como los filtros demográficos ya se encuentran implementados en la herramienta LigQ: para una proteína (el "usuario"), se buscan experimentos realizados con proteínas de la misma familia (información demográfica) y a partir de los ligandos obtenidos (los "items"), se buscan ligandos similares (basado en contenido). El resultado final es un filtro híbrido entre ambos, y sus distintas etapas se explicarán más en detalle en el chapter 4.

Filtros colaborativos basados en memoria

Inicialmente, al abordar el presente trabajo, exploramos la posibilidad de utilizar filtros colaborativos basados en memoria sobre las features de pockets y las de ligando. Planteamos que, al representar un pocket como una lista de los ligandos que se acoplan a éste, podríamos establecer la similitud entre un pocket y otro en función de dichas interacciones. Sin embargo, en nuestro caso de estudio parece muy difícil pensar en un filtro colaborativo de este tipo, dado que en el conjunto de datos del cual partimos la mayoría de pockets se acopla a muy pocos ligandos. Esto introduce en problema en la representación de cada pocket, dado que las métricas de similitud difícilmente permitirán encontrar pockets parecidos. Al respecto, surge como una posible alternativa el concepto de *pocketoids* detallado en el capítulo 2, sección 2.2.2.4. A diferencia de los pockets originales, en la sección mencionada se describe cómo los pocketoids exhiben una cantidad de interacciones más elevada, suficiente para aplicar un filtro colaborativo basado en memoria. Inclusive, al igual que se realiza en el curado de datasets de amplio conocimiento público como MovieLens[66], podría mejorarse aún más la calidad de los datos descartando aquellos pocketoids que tengan una cantidad de interacciones debajo de un cierto umbral (en inglés, *threshold*) a definir[a].
Si bien esta opción puede constituir una vía de investigación válida, se presenta sin embargo un problema intrínseco a este tipo técnicas en sí mismo: cuando estamos en presencia de un nuevo pocket, del cual no conocemos ninguna interacción con ligandos (el caso que justamente queremos atacar en este trabajo de tesis), no tenemos información con la cual realizar comparación y, en consecuencia, no podríamos dar ninguna recomendación. Este problema se conoce como *cold-start*[35] y es uno de las grandes desventajas de utilizar este tipo de métodos. Es la razón por la que decidimos explorar otras alternativas.

Filtros colaborativos basados en modelos

Los métodos basados en aprendizaje automático suelen dividirse en dos grandes familias: métodos de clasificación y métodos de regresión. Los algoritmos de clasificación se usan cuando el resultado deseado es una etiqueta discreta (usualmente denominada categoría o clase), mientras que los algoritmos de regresión buscan estimar un valor continuo.
Una segunda clasificación importante de los modelos es el modo en que se da el aprendizaje o entrenamiento del modelo. Existen, también, dos grandes familias de técnicas: las de aprendizaje supervisado, en donde uno le provee al algoritmo tanto elementos de entrada (input) como de salida (output), y no supervisado, en donde el algoritmo no cuenta previamente con los valores de la salida. En nuestro caso, nosotros contamos con la información de los acoplamientos de los pockets con ligandos y, usaremos esta información para entrenar al modelo. Por lo tanto, nos basaremos en técnicas de aprendizaje supervisado.
A continuación, analizamos algunos de los métodos que más se adaptan a nuestro problema.

Clasificación

Uno de los métodos más intuitivos de comprender es el de clasificación. En particular, definimos un problema como un problema de clasificación *binaria* cuando para un vector de entrada,

[a] Este dataset brinda información de usuarios, sus datos demográficos y sus valoraciones sobre un amplio listado de películas. Los datos se encuentran filtrados de tal manera que sólo se consideran los usuarios que hayan valorado al menos 20 películas.

el modelo debe asignarle una de dos categorías posibles. Si dichas categorías se refieren a si un elemento "pertenece" y "no pertenece" a un conjunto, podemos señalar a éstas como "caso positivo" y "caso negativo", respectivamente[67].
Nuestro problema puede pensarse como un problema de clasificación binaria en el que, dado un vector de entrada conformado por las features de un pocket y un ligando, debemos predecir si: ocurre acoplamiento (caso positivo) o no ocurre (caso negativo). Dado que buscamos predecir este acoplamiento para todos los ligandos, el algoritmo debería construir estos vectores para todas las combinaciones pocket-ligando[b], como describe el código de **Algorithm 1**.

Data: $M_{classifier}$: estimador entrenado
Data: F_p: features de pocket p
Result: IDs de los ligandos que potencialmente se pueden acoplar al pocket
forall *l in Ligandos* **do**
  $F_{combined} = F_p \mathbin{++} F_l$
  if $M_{classifier}.predict(F_{combined}) == Clase_{ocurre\_acoplamiento}$ **then**
    agregar el ID del ligando l a lista de resultados
  end
end

Algorithm 1: Ejemplo usando un clasificador para resolver el problema de Virtual Screening

Problemas de adoptar la clasificación binaria Si bien el problema de clasificación parece una buena opción, encontramos ciertos inconvenientes al utilizarlo para nuestro problema. En nuestro dataset figuran únicamente los casos de acoplamientos exitosos, es decir, faltan los casos negativos, lo cual es un componente vital para la técnica de clasificación. Ante esta situación, la mayoría de los métodos asumen que la ausencia de datos sugiere una ausencia de actividad[68], lo cual podría llevar a excluir erróneamente ligandos activos no conocidos. En este sentido, es importante notar que dada la especificidad de los experimentos (solventes utilizados, temperatura, etc.), cabe preguntarse si el acoplamiento no podría funcionar bajo otras condiciones. Es decir, no podemos asegurar que la ausencia de una unión pocket-ligando en el dataset implique que no puedan interactuar.

Para atacar el problema de la falta de casos negativos, existen bases de datos exclusivamente de decoys, *"moléculas computadas en base a propiedades físicas similares a las buscadas pero con diferencias en cuanto a su estructura química"*. Una de ellas es DUD-E (*A Database of Useful (Docking) Decoys: Enhanced*)[69]. Sin embargo, decidimos no utilizar esta base de datos por varias razones:

- **Asunción de inactividad**: dado que no se ha validado experimentalmente que los decoys sean inactivos, la posible presencia de compuestos activos en el conjunto de decoys puede introducir una subestimación artificial del enriquecimiento del conjunto original[70].
- **Sesgo experimental al incorporarlos:** el volumen de datos que aporta, referente a proteínas con sus acoplamientos, es mínimo en relación a los existentes en el dataset

[b] Este tipo procesamiento debe prestar especial atención al uso de las capacidades de procesamiento en paralelo que provea el hardware en que se ejecute el código debido a la gran cantidad de combinaciones pocket-ligando de nuestro dataset (del orden de millones).

con el que contamos. DUD-E contiene 102 proteínas, mientras que nuestro dataset tiene alrededor de 3.000 (es decir, menos del 4% del total). De esta forma, el dataset final quedaría severamente desbalanceado, lo que produciría que cualquier clasificador presente un sesgo (en inglés, *bias*) en favor de la clase que se encuentra sobre-representada. Una de las alternativas para evitar este problema es o bien un up-sampling de las clases sub-representadas o un down-sampling de las clases sobre-representadas. Para más información ver Trabajo Futuro[71].

- **Feature faltante**: los compuestos de DUD-E provienen de la base de datos ZINC, la cual no incluye la propiedad "número de anillos aromáticos" (`arom_rings` en el dataset). De forma que habría que hacer un nuevo cálculo para estos casos.

Debido a los argumentos expuestos anteriormente, la técnica de clasificación binaria tradicional se vuelve inaplicable en nuestro problema. Si bien existen alternativas[c] que pueden plantearse en este escenario, decidimos descartar la técnica en función de una que creemos más apropiada para nuestro caso.

Regresión

Si la clasificación tiene la tarea de asignar una clase, es decir, predecir a qué clase pertenece un conjunto de datos (aquí es muy importante entender que en los problemas de clasificación los valores son discretos), la regresión tiene el objetivo de predecir valores continuos. En nuestro caso particular, en donde prácticamente todos los descriptores están representados por número reales, podemos plantear el problema de estimar los valores que tiene que tener cada feature de ligando para corresponderse con las features de un pocket. Esto permite pensar en un circuito que describiremos en la sección siguiente, y que fue por el que finalmente optamos.

Pipeline propuesto

El circuito que proponemos fue inspirado en trabajos previos del área[47], y comprende dos etapas. En el primer paso, a partir de la información del dataset entero (`unified_pockets.csv`, `bindings.csv` y `reduced_ligands.csv`), se utiliza un modelo que recibe como entrada los descriptores de un pocket y predice las características que debe tener un compuesto para poder acoplarse a éste. Como existen altas probabilidades de que ningún compuesto del dataset cuente exactamente con estos valores, nos referiremos a este compuesto como el ligando *ideal* para el pocket.
Para armar el conjunto de entrenamiento con el que se contruye el modelo, realizamos la junta entre la tabla de features de pockets y la de features de ligandos usando la tabla de acoplamientos como nexo, excluyendo los ligandos que quedaron sin participar en alguna interacción. De este modo, obtuvimos un listado de pares *features de pocket−features de ligando*. A partir de este conjunto, entrenamos el modelo de regresión y optimizamos los hiperparámetros, quedándonos con el mejor modelo obtenido. El mismo fue guardado en un archivo de tipo `.joblib`, pudiendo de esta manera ser utilizado en distintas aplicaciones.
En la segunda etapa, utilizando un modelo de clustering sobre los datos de `ligands.csv`, evaluamos en cuál de los conjuntos se ubica el ligando *ideal*, y elegimos como ligandos candidatos

[c] Una opción es definir nuestro problema como un problema de Positive-Undefined Learning (abreviado *PU learning*)[68]. Para más detalle de esta técnica consultar el Apéndice A, subsection A.3.1

a los compuestos de dicho cluster que cumplen ciertas condiciones. En particular, usando la distancia euclídea entre vectores, realizamos la búsqueda de ligandos similares tomando los n más cercanos. Para la construcción del modelo de clustering usamos la técnica K-means y, a fin de elegir el valor de la cantidad de clusters (k), hicimos variar dicho parámetro y graficamos la inercia del clustering total. En la figura 3.1 se puede observar que alrededor de $k = 200$ la curva pasa de un descenso no-lineal a uno lineal, donde varía mucho más levemente. A modo de comparación, realizamos el mismo análisis para los ligandos presentes en `reduced_ligands.csv`, y los resultados de la figura 3.2 fueron muy similares. Más allá de los valores de la inercia, que pueden variar de acuerdo a la cantidad de elementos y su distancia, alrededor de $k = 200$ la curva muestra un una forma similar al caso de `ligands.csv`, de modo que en ambos casos, a partir de ese punto, cualquier valor de clustering elegido parece ser una buena alternativa.

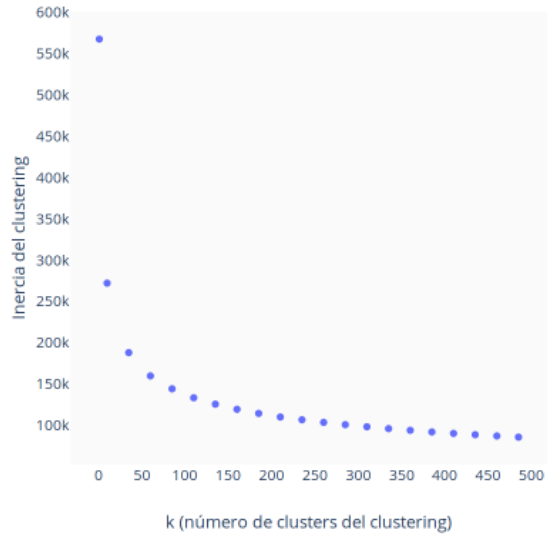


Figure 3.1: Medida de inercia del clustering de `ligands.csv` a medida que aumentamos el parámetro k de K-means.

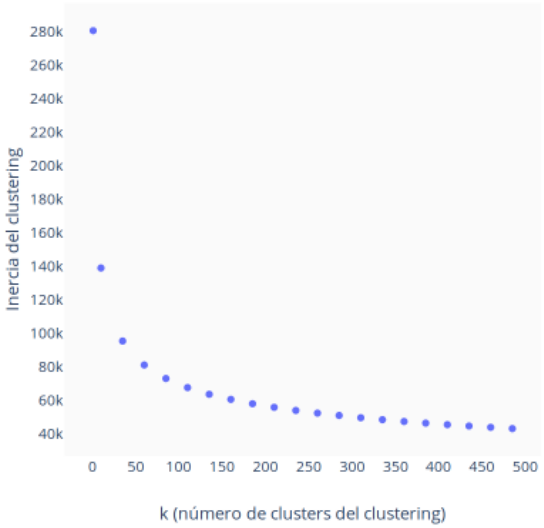


Figure 3.2: Medida de inercia del clustering de `reduced_ligands.csv` a medida que aumentamos el parámetro k de K-means.

De la misma manera que para el modelo de regresión de la primera etapa, en este caso también se persiste el modelo de clustering en un archivo de tipo `.joblib` para futuros usos, a la vez que generamos para cada cluster una estructura denominada BallTree, que permite obtener distancias entre elementos de manera eficiente[d].
El pipeline completo puede visualizarse en la figura 3.3.

[d] A nivel implementativo, utilizamos un BallTree para cada cluster. Esta estructura nos permite realizar la búsqueda de proximidad de forma eficiente.Para más detalle de esta técnica, consultar el Apéndice A, sección A.3.6.

Figure 3.3: En la etapa A, para cada uno de los pockets identificados en la proteína, se calcula el ligando *ideal* que mejor chance tiene de acoplarse al mismo. En la etapa B se buscan mediante clustering aquellos ligandos del dataset completo (archivo `ligands.csv`) más similares al ligando ideal.

El ligando *ideal* puede inclusive resultar interesante para posteriores estudios, dado que podría buscarse sintetizar un compuesto aún no existente que cumpla con dichas características. [72] [73] [74] [75]

3.2 Metodología

En esta sección ahondaremos en los detalles de las técnicas de aprendizaje automático que utilizamos en nuestro trabajo. Explicaremos, entre otras cosas, por qué es necesario realizar un escalado de los valores de las features de entrada y salida, y comentaremos la metodología seguida para realizar el entrenamiento del modelo y, mencionaremos cómo y cuáles son las formas en que se puede evaluar el rendimiento de los algoritmos utilizados.

3.2.1 Escalado y transformación de features

Con el fin de reducir sesgos y mejorar la performance de los modelos de aprendizaje automático, en la literatura [76] se sugiere que los datos de entrenamiento y predicción deben poseer ciertas propiedades estadísticas a saber:

- De ser posible, las variables predictoras (también llamadas "independientes") no deben estar correlacionadas.
- Todas las features deben tener una media similar para evitar durante el entrenamiento el sesgo hacia un valor determinado. También deben tener la misma varianza, para disminuir la probabilidad de que una variable cobre mayor relevancia que las demás. Se sugiere que la media de cada variable sea 0 y la varianza 1[e] porque suele mejorar la capacidad de generalización de algoritmos de aprendizaje automático, como SVM [77].
- El dominio de las variables debe encontrarse en el rango [0, 1], o en el [-1, 1]. En el caso de los datos de entrada de una red neuronal (descripta a continuación en la sección 3.3.2), ésto es para incrementar la probabilidad de que la salida de una capa se ubique en ese rango. En el caso de los datos de salida, esto tiene como objetivo que dichos datos sean consistentes con el rango de la funciones de activación sigmoideas (mencionadas en sección 3.3.2, cuando introducimos el concepto de red neuronal).

[e] Usualmente por medio de un procedimiento denominado "normalización Z-score" (comúnmente, sólo "normalización"), que consiste en sustraer la media de cada feature y dividir por el desvío estándar en todas las muestras. En inglés se usa también el término *standardization*.

- Si el valor de una variable no se encuentra en los rangos mencionados, se sugiere escalarla aplicando una transformación robusta, es decir, tolerante a *outliers*. Esto se puede lograr, por ejemplo, mediante el análisis de cuantiles para dispersar los valores concentrados y agrupar los valores extremos o más distantes.

- La transformación de variables no debe alterar las relaciones subyacentes entre las features, en especial, al entrenar el modelo de regresión y realizar predicciones. Por ejemplo, se debe evitar escalar sólamente una de las features.

Considerando las propuestas anteriores, decidimos realizar una transformación robusta de la distribución de las propiedades de pockets y de ligandos en el rango [-1, 1]. Para su procesamiento, utilizamos dos clases auxiliares que ofrece scikit-learn y las aplicamos de forma secuencial: en primer lugar `QuantileTransformer` y luego `MinMaxScaler`. La primera implementa una transformación no lineal del conjunto de datos (cada feature de manera independiente), de forma tal que siga una distribución uniforme en el intervalo [0, 1]. Para ello, agrupa los datos por cuantiles, consiguiendo a la vez a) espaciar los valores más frecuentes y b) reducir el impacto de valores outliers. La clase `MinMaxScaler` implementa, en cambio, una transformación lineal que toma como parámetros los nuevos valores de mínimo (`min`) y máximo (`max`) (la fórmula se detalla en la sección A.3.2). En nuestro caso, los valores utilizados fueron [-1 y 1], respectivamente. De este modo, además conseguimos que la distribución de los datos de cada feature tengan media 0 y varianza 1.
Ambas clases también permiten realizar la transformación inversa, es decir, el desescalado. Para ello creamos una clase llamada `ChainedPreprocessor`, que almacena las dos transformaciones sobre un conjunto específico de datos.
Es importante mencionar que las propiedades de nuestros ligandos se encuentran en dos archivos: `ligands.csv` (todos los ligandos) y `reduced_ligands.csv` (sólo ligandos que se acoplan a pockets dentro de nuestro dataset). Cada uno de estos archivos puede seguir distribuciones de datos diferentes. De esta manera, las respectivas transformaciones T_{total} y T_{sub} pueden diferir. Por esta razón evaluamos dos alternativas distintas de escalado de datos:

- **Dos escalados**: escalar los ligandos de `reduced_ligands.csv` con T_{sub} y los de `ligands.csv` con T_{total}. Esto implica que el ligando ideal obtenido como salida de la etapa A debe ser desescalado, y reescalado con T_{total} para usarlo como entrada de la etapa B del pipeline (ver figura 3.3).

- **Único escalado**: escalar tanto los ligandos de `reduced_ligands.csv` como los de `ligands.csv` con T_{total}. Descartamos usar T_{sub} en ambos porque podría hacernos perder acceso a ciertos valores de la distribución de datos de `ligands.csv` que son poco frecuentes en `reduced_ligands.csv`, sobre todo los extremos.

Los diferentes escalados mencionados son aplicados durante el entrenamiento de cada modelo según corresponda.

3.2.2 Muestra de control

Sobre el conjunto de datos ya curado tomamos una muestra "balanceada" correspondiente a apróximadamente el 10% de los datos (muestra de control) y la dejamos completamente aislada de los datos de entrenamiento del modelo de aprendizaje automático. Establecimos la restricción

de que todos los ligandos asociados a los pockets de la partición también debían estar incluidos. Para conseguir este objetivo con el dataset 8L, utilizamos la clase `GroupShuffleSplit` de scikit-learn, la cual selecciona aleatoriamente elementos en la proporción indicada a la vez que mantiene juntos aquellos que pertenecen a un mismo grupo. En nuestro caso tomamos como etiqueta de grupo el pocket_id. En cuanto al dataset sesgado, no era posible conseguir en simultáneo una muestra balanceada y que todos los ligandos de un pocket queden incluidos debido a los casos atípicos, por lo que terminamos utilizando el método `train_test_split` (basado en la clase `ShuffleSplit`), para extraer el 10% sin dicha restricción.
Durante la etapa de entrenamiento y validación, utilizamos técnicas de *cross-validation* (ver sección 3.2.4) para ajustar el modelo. Una vez entrenado cada modelo de regresión, la muestra de control fue utilizada una única vez, como forma de medir la performance del pipeline completo.

Para realizar un "doble chequeo" a la hora de evaluar la performance del pipeline de recomendación ya con la configuración final, creamos una muestra de control adicional dentro de los datos que inicialmente habían sido seleccionados para el entrenamiento del modelo, quedando globalmente, sobre el 100% de los datos:

- 80% (**train & validation**): datos de entrenamiento y cross-validation
- 10% (**test**): primer grupo de control
- 10% (**test**): segundo grupo de control

Al momento de particionar el conjunto de datos en training y test, tomamos los recaudos necesarios para minimizar el problema de ***dataset shift***[78] ("variación/desplazamiento en el conjunto de datos") o *drifting*. Este problema refiere a la diferencia en la distribución de los datos entre particiones. La presencia de un desplazamiento puede introducir un sesgo en el entrenamiento y podría perjudicar la capacidad de generalización del modelo[f]. A continuación, detallaremos tres tipos de dataset shift[79]: *covariate shift*, *prior probability shift* y *concept shift*.

Covariate Shift

El problema del *covariate shift* se refiere al cambio en la distribución de las variables de entrada presentes en las muestras de entrenamiento y de control. Si existe, al unir ambas particiones, debería ser posible identificar si un registro pertenece a uno u otro grupo con suficiente precisión. Una forma de detectar esto es agregar un nuevo campo `origin` que indique a qué partición pertenece, por ejemplo 0 si es train y 1 si es test. Luego se debe entrenar un modelo de clasificación binaria (que tenga como entrada una feature por vez) con una porción balanceada de este campo. Luego, se debe calcular el valor ROC AUC[80] utilizando la predicción sobre el grupo restante. Si para alguna feature el valor obtenido es mayor que un umbral predefinido, por ejemplo 0.8, decimos que existe un desplazamiento para esa variable.[g]
Decidimos implementar el procedimiento anterior para nuestros datos, y nos encontramos con que el valor de ROC AUC para cada una de las features de `pockets.csv` era de aproximadamente 0.5 (equivalente a una selección aleatoria), tanto en el conjunto de datos de entrenamiento como en el de evaluación. De esta manera, concluímos que no es posible distinguir, a partir de las variables de entrada, de qué partición proviene una muestra.

[f] Realizamos la partición y validación del conjunto de datos utilizando los dos esquemas de escalado mencionados en la sección 3.2.1. Los mismos se encuentran almacenados en directorios independientes.

[g] `https://www.kaggle.com/pavansanagapati/tutorial-covariate-shift-sberbank-housing-eda`

Prior Probability Shift

El problema de *prior probability shift* es equivalente al covariate shift, pero sobre el dominio de las variables de salida. Para analizar si nuestras particiones del dataset se ven afectadas por este problema, repetimos el mismo procedimiento de covariate shift. Como resultado, obtuvimos que para todas las features de salida, sus valores de ROC AUC fueron cercanos a 0.5, es decir, equivalentes a una selección aleatoria y, por lo tanto, pudimos concluir la ausencia del problema prior probability shift.

Concept shift

Finalmente, el problema de *concept shift* sucede cuando se produce una alteración en la relación de dependiencia entre las variables. No es un problema trivial de detectar, así que optamos por comprobar que no exista *dataset shift*, evaluando únicamente la ausencia de covariate shift y prior probability shift.

3.2.3 Métricas

Definir una métrica tiene como fin último poder evaluar qué tan preciso o "bueno" es un modelo y, finalmente, permite obtener una conclusión acerca de cuál de todos elegir.
Para el caso de los problemas de regresión suele definirse lo que se conoce como **función de pérdida** (en inglés, *loss function*), que mide cuánto se alejan los valores predichos de los valores reales. El objetivo es minimizar dicha función. Las funciones de pérdida más utilizadas son: R^2, *error medio absoluto*, *error cuadrático medio* y *raíz del error cuadrático medio*. Las ecuaciones de dichas funciones, junto con sus principales características, pueden consultarse en el Apéndice A, sección A.3.3.
Para los problemas de clustering existen distintos tipos de métricas. En particular ya hemos mencionado una de ellas en el capítulo Confección del Dataset: la inercia (ver Subsección 2.2.2). Esta métrica mide el grado de cohesión interna de los clústers, y su fórmula puede encontrarse en el Apéndice A, subsection A.3.4. También existen otras métricas, como el *silhouette score* [81] (promedio), que mide tanto la distancia intra-cluster (grado de cohesión) como la inter-cluster (grado de separación entre ellos). Hicimos algunas pruebas con esta última[h] pero posteriormente la descartamos en favor de la inercia por cuestiones de performance computacional, dado que requiere precalcular la distancia entre cada par del total de elementos (todos los pockets entre sí).

3.2.4 Cross-validation

Una manera posible de evaluar un modelo es tomando los datos originales y crear a partir de ellos dos conjuntos separados: un primer conjunto de entrenamiento y un segundo conjunto de validación. De esta forma, podemos identificar si el modelo sufre del fenómeno conocido como "sobreajuste" u *overfitting*, por su denominación en inglés. Esto ocurre cuando el modelo conoce "demasiado" bien los datos de entrenamiento (inclusive el ruido y las anomalías de éstos), impidiendo un verdadero aprendizaje, conllevando a una memorización extrema de los datos (especificidad), e imposibilitando obtener buenas estimaciones para otros casos (generalización).

[h] Utilizamos los servicios computacionales que nos proporcionó el Centro de Computación de Alto Rendimiento (CeCAR), una dependencia de la Facultad de Ciencias Exactas y Naturales de la UBA. Para más información, su sitio web es: `https://cecar.fcen.uba.ar/`

Dada la naturaleza aleatoria en la construcción de los conjuntos de entrenamiento y validación, la performance del modelo puede variar dependiendo de las características de los datos presentes en cada una de las particiones. Es por ello que, para obtener los mejores resultados en cuanto a la medición de la calidad de la predicción de los modelos, es preferible realizar una **validación cruzada** o *k-fold cross validation* sobre los datos de entrada. Esta técnica entrena de manera iterativa cada modelo sobre todo el conjunto de entrenamiento tomando k particiones de éste último. Al momento de realizar el entrenamiento, se va a tomar cada partición k como conjunto de prueba del modelo, mientras que el resto de los datos se usarán para el entrenamiento. Este proceso se repetirá k veces, y en cada iteración se seleccionará un conjunto de prueba diferente, mientras los datos restantes se emplearán, como se mencionó, como conjunto de entrenamiento. Una vez finalizadas las iteraciones, se calcula la precisión y el error para cada uno de los modelos generados, y para obtener la precisión y el error final se calcula el promedio de los k modelos entrenados, y se seleccionará al final aquel que produzca el mejor valor de precisión y menor error promedio.
Finalmente, confiamos en que el resultado en promedio nos de una mejor medida de cuál es el desempeño del modelo de clasificación sobre el conjunto entero de datos.

3.2.5 Selección de hiperparámetros

Los hiperparámetros conforman aquellas configuraciones del modelo que uno elige antes de que el proceso de aprendizaje comience. Por ejemplo, para el caso de una red neuronal pueden ser la arquitectura de la red (cantidad de capas y sus densidades), la tasa de aprendizaje, o bien la función de activación de cada neurona. La adecuada selección de estos valores determinará si el modelo logra acercarse a los valores de predicción óptimos. Uno de los algoritmos más utilizados para el ajuste de hiperparámetros, y por el cual optamos, es el de **"búsqueda en rejilla"** (más conocido como *Grid Search*). Esta técnica permite comparar, de forma completamente automatizada, distintas combinaciones predefinidas de hiperparámetros a través de métricas de rendimiento de modelos. Si bien esta combinatoria exige un cálculo oneroso, la compensación es, hipotéticamente, obtener el mejor modelo posible y una mayor garantía de poder reproducir el experimento. En la figura 3.4 se detalla el proceso global de Grid Search.

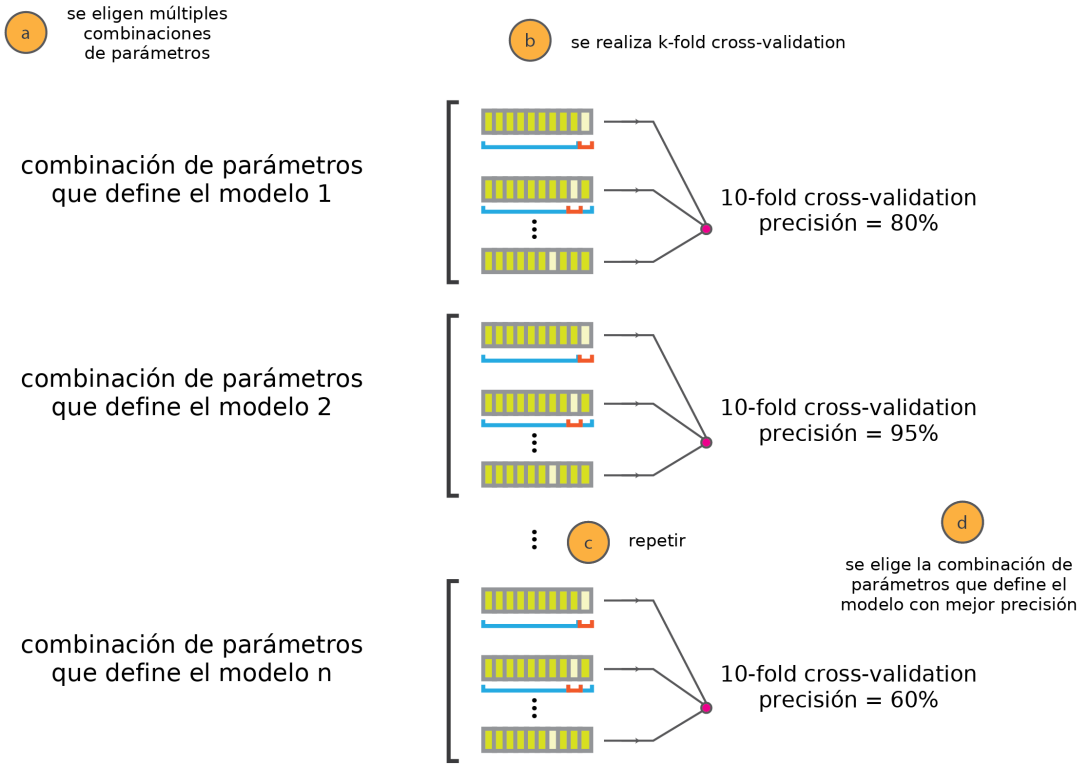


Figure 3.4: A la izquiera del gráfico (a) se encuentra la etapa en donde se construyen las distintas combinaciones de hiperparámetros. En la segunda etapa (b) se utiliza un esquema de validación cruzada para medir la performance del modelo. Este procedimiento se repite con cada una de las combinaciones de parámeteros (c). Finalmente en la etapa (d) se elige el modelo con mejor puntaje. Fuente: adaptado de `https://cambridgecoding.wordpress.com/2016/04/03/scanning-hyperspace-how-to-tune-machine-learning-models/`

La desventaja que presenta este método es que si la granularidad de la rejilla no es lo suficientemente fina puede ser dificil identificar si la función objetivo del modelo queda atrapada en mínimos locales[82]. Existen esquemas alternativos similares como la **optimización aleatoria** (en inglés, *Random Search*), en donde se prueban un número fijo de valores de hiperparámetros que surgen de una distribución elegida por el usuario. El número de valores probados está dado generalmente por la cantidad de iteraciones elegida. Decidimos no utilizarlo, en favor de que los resultados sean repetibles.

3.3 Predicción del ligando *ideal*

Para predecir el ligando ideal (ver figura 3.3, etapa A de nuestro pipeline), exploramos dos métodos basados en técnicas de regresión: **vecinos más cercanos** (en inglés, *Nearest Neighbours*) y **redes neuronales** (en inglés, *Neural Networks* o NN). A continuación detallaremos cada uno de ellos.

3.3.1 Vecinos Más Cercanos aplicado a regresión

En la sección 2.2.2.1 vimos, a partir de las figuras 2.1 y 2.2, que a medida que los pockets se parecen más entre sí, sus ligandos asociados también se parecen mucho entre sí (mismo comportamiento). En base a este resultado, la técnica de vecinos más cercanos aplicada a regresión puede considerarse como una opción interesante. A su vez, esta técnica suele funcionar bien cuando la cantidad de variables de entrada no es demasiado grande, como en nuestro caso (16 features para pockets).
El principio detrás del método de vecinos más cercanos es el siguiente: dado un nuevo elemento, se encuentran aquellos más cercanos al mismo (vecinos) y, finalmente se devuelven sus valores interpolados. En problemas de regresión, se suele calcular el valor medio de cada feature. El algoritmo requiere, o bien que se defina el número k de elementos más cercanos a devolver (k-NN) o un radio fijo dentro del cuál buscar los vecinos. En nuestro caso, decidimos utilizar el primero, con la implementación de scikit-learn (`KNeighborsRegressor`), variando la cantidad de vecinos (parámetro k) y midiendo los resultados otorgados por un esquema de cross-validation *10-fold*.
Es importante mencionar que el algoritmo funciona mucho mejor si todos los datos tienen la misma escala, razón por la cual utilizamos los escaladores definidos anteriormente.
Los modelos k-NN suelen involucrar la configuración de múltiples parámetros: el número de vecinos, la métrica de distancia y la distribución de los pesos iniciales, entre otros. La distancia puede, en general, ser cualquier medida. En nuestro caso, decidimos utilizar la distancia *Manhattan*[i] (debido a que presenta buenas propiedades ante la presencia de outliers [83]), una distribución uniforme de los pesos y una cantidad variable de vecinos, a partir de la cual generamos múltiples modelos. Para medir la performance del modelo, la métrica que decidimos utilizar fue MSE (Mean Squared Error) y, como detalle, mencionamos que `scikit-learn` requiere configurarla con el parámetro `greater_is_better=False`, dado que la misma es una función de pérdida y se busca minimizarla. A nivel implementativo esto hace que la métrica tome valores invertidos en signo, es decir, valores negativos. Como toda función de pérdida, tomaremos como resultados más exitosos a aquellos cuyos valores se encuentren lo más cercanos al 0 posible. En este sentido, cabe aclarar que la métrica por sí sola no tiene una interpretación "correcta", sino que es útil para comparar la performance de varios modelos, o para medir el error en función de la "experiencia" (detallado más adelante). Por otro lado, inclusive si ocurriese que la métrica resulta en un valor de 0, esto no significa que nos encontramos ante un modelo perfecto, dado que puede que sea perfecto para los datos de entrenamiento pero no para los datos de validación, dando lugar al fenómeno de *overfitting*, ya mencionado anteriormente.
En nuestro caso, nos enfocamos en analizar cómo varía el rendimiento de los modelos a medida que aumenta el número de vecinos del algoritmo k-NN. Los resultados pueden verse en la figura 3.5, en donde se muestra la media (punto) y el desvío estándar (barras verticales) del MSE de los resultados en la validación cruzada 10-fold. Este experimento no involucra datos de ChEMBL, sino que fue realizado exclusivamente con el dataset de PDB.

[i] La utilización de otras distancias es discutida en el capítulo 8 (Trabajo Futuro).

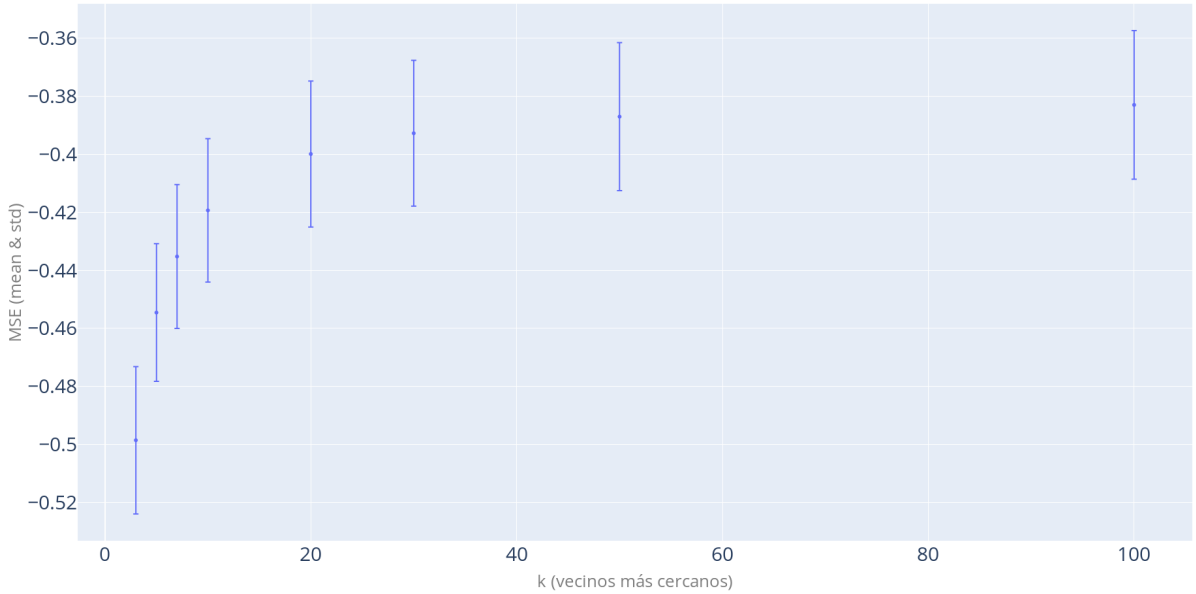


Figure 3.5: Error cuadrático medio para distintos valores del parámetro k del método de vecinos más cercanos. Para cada k se promediaron los resultados de la validación cruzada de 10-fold. Las barras se corresponden con el desvío estándar.

Podemos ver que a medida que se incrementa el valor de k el error tiende a acercarse más a 0, lo cual es un indicio de mejora en la performance del modelo. Sin embargo, a partir de un valor de k de 50 el error cuadrático medio tiende a converger a -0.37. Esto marca un umbral en el ajuste del parámetro k del modelo. Si bien existe la posibilidad de que con valores más altos se lleguen a otros resultados, corridas hechas con $k = 100$ y $k = 500$ no mostraron mejoría alguna. Aunque el estudio del error cuadrático medio puede dar un indicio de qué valor de k seleccionar para construir el mejor modelo, tomando por ejemplo el punto de quiebre de la curva, es necesario describir las características del modelo en términos de su capacidad de generalización. Para ello, graficamos la **curva de aprendizaje** (en inglés, *learning curve*[j]) en función de una cantidad creciente de muestras (figuras 3.6 a 3.13). Como parámetros, tomamos un esquema de cross-validation con 10 iteraciones, cada vez con 70% de los datos como entrenamiento y 30% como validación, seleccionados de forma aleatoria. En rojo representamos los puntajes correspondientes al error promedio para los casos de entrenamiento y, análogamente, en verde el error promedio para los casos de validación. Las áreas en rojo y verde reflejan el desvío estándar, sin embargo, debido a que dichos valores son muy pequeños, no pueden ser apreciados en los gráficos.

[j] Utilizamos la implementación propuesta por `sci-kit learn` en `https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html` con una pequeña modificación para poder incorporar la métrica MSE.

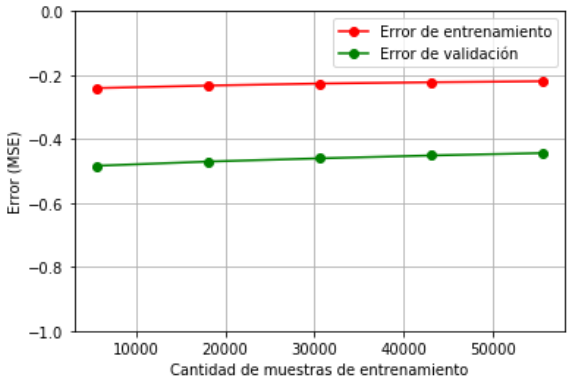


Figure 3.6: Curvas de aprendizaje para un modelo KNN con k igual a 3.

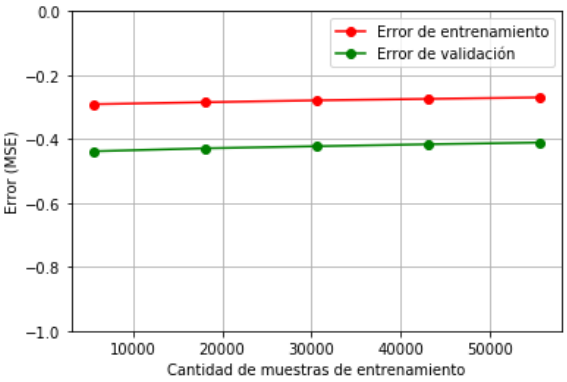


Figure 3.7: Curvas de aprendizaje para un modelo KNN con k igual a 5.

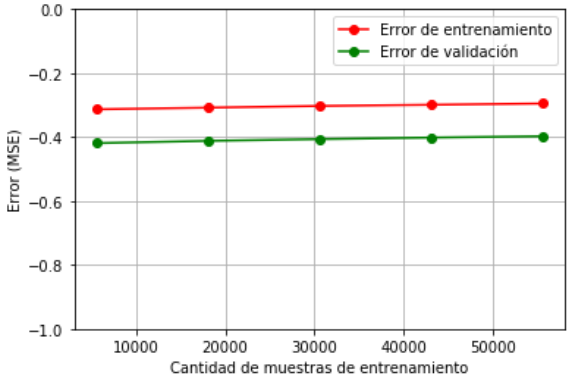


Figure 3.8: Curvas de aprendizaje para un modelo KNN con k igual a 7.

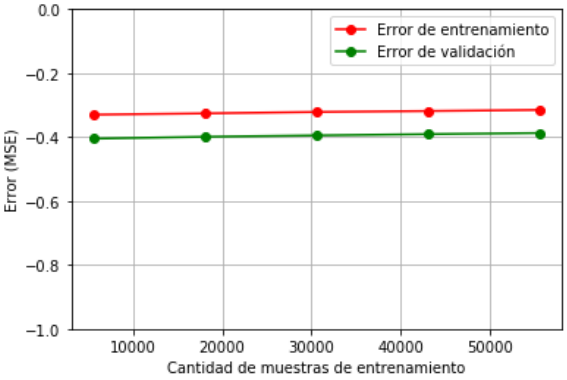


Figure 3.9: Curvas de aprendizaje para un modelo KNN con k igual a 10.

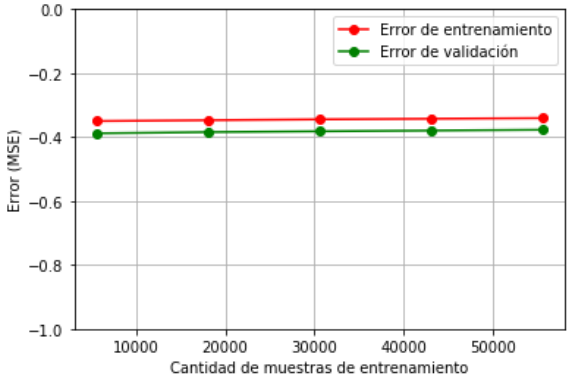


Figure 3.10: Curvas de aprendizaje para un modelo KNN con k igual a 20.

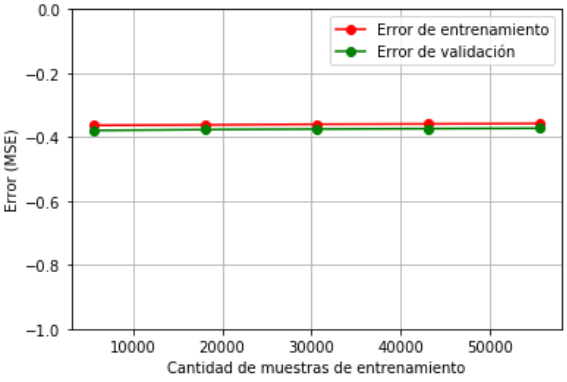


Figure 3.11: Curvas de aprendizaje para un modelo KNN con k igual a 50.

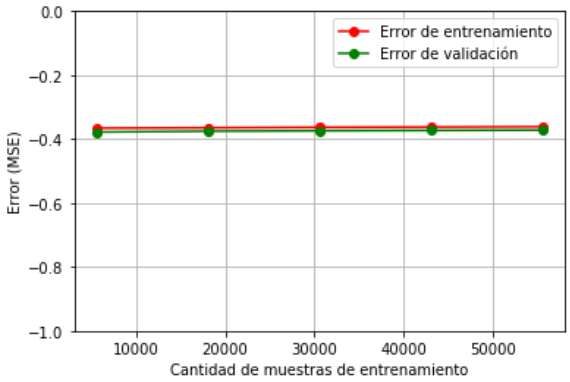


Figure 3.12: Curvas de aprendizaje para un modelo KNN con k igual a 70.

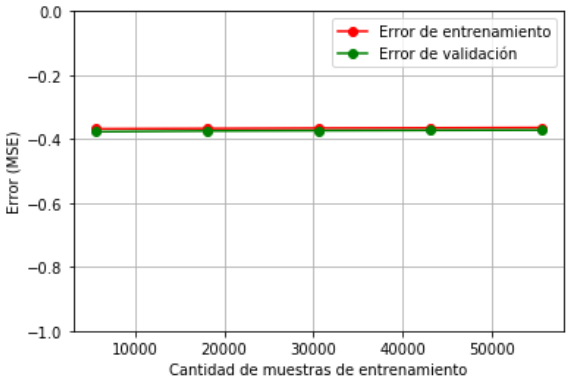


Figure 3.13: Curvas de aprendizaje para un modelo KNN con k igual a 100.

En todos los gráficos (figuras 3.6 a 3.13) observamos el mismo comportamiento que notamos en la figura 3.5: a medida que incrementamos el valor de k el error disminuye. También se visualiza que, a medida que aumenta el valor de k, la diferencia entre los valores de error cuadrático medio de entrenamiento y validación tienden a achicarse, lo cual demuestra que a medida que incorporamos más vecinos, el modelo generaliza mejor. Sin embargo, observamos que el error cuadrático medio prácticamente no varía con el aumento de muestras, lo cual nos indica que el modelo no es capaz de aprender de los casos de entrenamiento. En este caso, se dice que el modelo sufre de una sub-estimación (fenómeno conocido como *underfitting*). Dicho de otra manera, el modelo no puede capturar la complejidad del dataset de entrada, de manera que no importa la cantidad de muestras que se le presenten, la performance se mantiene prácticamente constante.
Para atacar dicho problema, podemos o bien variar las features que tomamos o bien modificar los parámetros del algoritmo. Respecto a esta última alternativa, nos encontramos con que la técnica de vecinos más cercanos no tiene demasiadas opciones de ajuste como sí la tienen otras herramientas, como por ejemplo las redes neuronales. Adicionalmente, nos autoimpusimos la restricción de trabajar sólo con datos que maneja LigQ, de manera que la opción de incorporar nuevas features de pocket y/o ligando quedó fuera del alcance de este trabajo. Es por ello que, antes de seguir profundizando en esta técnica, decidimos probar con otras alternativas, las cuales se detallan a continuación

3.3.2 Redes Neuronales aplicadas a regresión

Otra manera de obtener un ligando ideal, a partir de las características de un pocket, es utilizando redes neuronales. Una **red neuronal artificial** (en inglés, *Artificial Neural Network* o *ANN*) es un modelo computacional inspirado en el sistema nervioso de los organismos animales. Comprende una estructura de nodos interconectados entre sí, a los que denominaremos "neuronas". Estas neuronas se organizan en capas y operan como lo haría una neurona real. Se utilizan conceptos como a) las **dendritas**, prolongaciones ramificadas de la neurona a través de las cuales recibe información procedente de otras, es decir, la entrada (*input*) y b) el **axón**, una fibra nerviosa que permite enviar las señales eléctricas a otras neuronas, es decir, la salida (*output*). A nivel formal, puede definirse como un algoritmo que estima una función de $f(\cdot) : R^m \rightarrow R^n$ a partir de un entrenamiento con un dataset, donde m es el número de dimensiones de la entrada y n es el número de dimensiones de la salida.

Existen diversos tipos de redes neuronales artificiales y, una de las principales diferencias entre ellas, es la organización de esta red, es decir, cómo se conectan las neuronas. En nuestro caso, estamos interesados en un tipo particular: el **Perceptrón Multicapa** (en inglés, *Multi Layer Perceptron*[84] o **MLP**). Un perceptrón es un modelo matemático que nos sirve para emular una neurona disparando una señal en función de los estímulos que recibe, y da nombre a uno de los modelos más conocidos para encarar un problema de regresión utilizando aprendizaje supervisado. Éste comprende una capa de entrada (conjunto de neuronas que representan cada una de las features de entrada), una o varias capas intermedias (también llamadas capas *ocultas*) y una capa de salida que representa la salida, como puede verse con distintos colores para cada una de ellas en la figura 3.14.

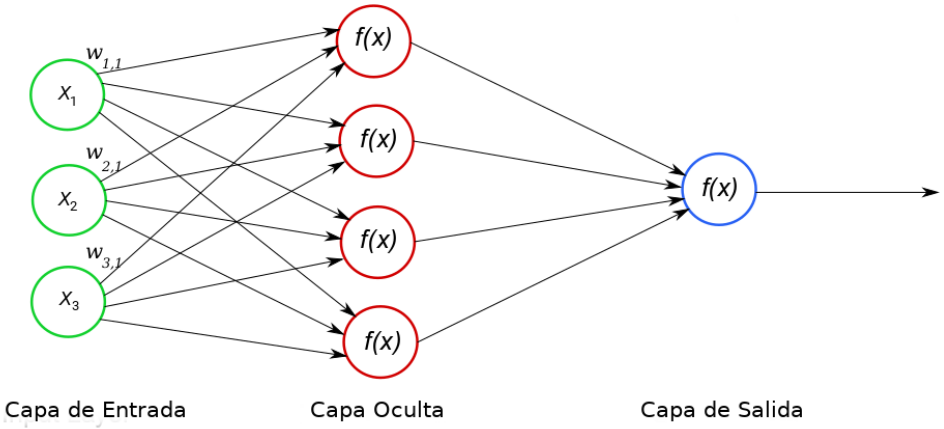


Figure 3.14: Representación gráfica de un único perceptrón con 3 entradas (verde), una capa oculta con 4 neuronas (rojo) y una salida (azul). Los pesos asignados a las conexiones que van desde las neuronas de entrada a la primera neurona de la capa oculta son $w_{1,1}$, $w_{2,1}$ y $w_{3,1}$. La salida, $f(x)$, de esta última es la aplicación de la función de activación a la suma lineal ponderada $w_{1,1}x_1 + w_{2,1}x_2 + w_{3,1}x_3$, esto es, una combinación lineal de las entradas y sus pesos.

La capa de entrada recibe por sus dendritas la entrada de la red y sus salidas alimentan a la primera capa oculta. Cada capa oculta, en caso de haber más de una, recibe sus entradas de las salidas de la capa anterior, y alimenta a la siguiente. Finalmente, la capa de salida recibe sus entradas de la última capa oculta y el resultado de las funciones de activación de cada neurona en esta capa es considerado el resultado del modelo. La intensidad de la conexión entre una neurona i de una capa y otra j de la siguiente se denomina **peso**, y se representa con un número real (denotado w)[k]. Todas las entradas junto con sus pesos se combinan linealmente obteniendo así el valor de entrada de la neurona. Luego, la neurona j aplica su función de activación, cuyo resultado es lo que el perceptrón computa como salida.

[k] La notación $w_{i,j}$ representa el peso de la conexión entre la neurona i con la neurona j de la capa siguiente. Por simplicidad, se omite j en el contexto de una neurona de llegada aislada.

En resumen, cada neurona toma múltiples entradas (estímulos) y computa a partir de ellas una salida (respuesta), utilizando, como ya mencionamos, una función de activación elegida para este propósito. Se trata de una función matemática no-lineal, usualmente en forma de "S" o sigmoide que se encuentra acotada tanto superior como inferiormente (ver figura 3.15). Existen múltiples funciones sigmoides: la función logística, la función logística generalizada, la arcotangente, la tangente hiperbólica, la función error o la función Gompertz son algunas de ellas.

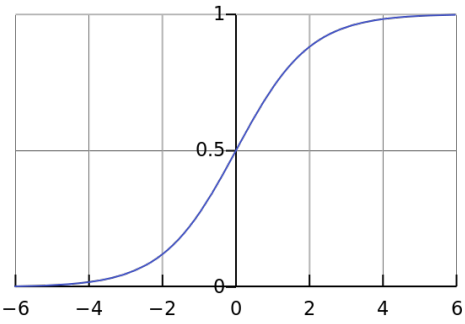


Figure 3.15: Ejemplo de función sigmoide

Dado el acotado rango de variabilidad de este tipo de funciones (usualmente alrededor de cero y, aproximándose a sus valores mínimo y máximo a medida que la entrada tiende a infinito), el perceptrón presenta poca posibilidad de adaptación cuando las entradas son valores muy grandes o muy chicos, dado que saturan el valor de salida de la neurona de destino e impiden ajustar correctamente al valor ideal deseado. Por este motivo se recomienda mantener los valores de entrada dentro de un rango pequeño mediante alguna técnica de escalado.[76]

Por otro lado, los pesos w regulan la importancia de cada entrada con respecto al valor de salida de la neurona, de forma que pasa a ser un parámetro de ajuste del modelo. Entrenar la red neuronal es, básicamente, ajustar los pesos de sus entradas de forma de minimizar la función de pérdida elegida (como fue mencionado en la sección 3.2.3). En este sentido, desde el punto de vista computacional, una red neuronal puede ser considerada como un algoritmo en donde ciertos parámetros numéricos (los pesos) no son especificados por el programador, sino calculados durante la fase de entrenamiento. La razón por la que los pesos no pueden ser especificados con anterioridad es porque usualmente no disponemos del conocimiento de la relación causa-efecto, es decir, cómo los diferentes valores de los pesos afectan el proceso computacional de la red.

Una manera de obtener este ajuste es mediante el algoritmo de **propagación hacia atrás de errores** o **retropropagación** (en inglés, *backpropagation*). Más precisamente, se entrena la red usando algún algoritmo de descenso por gradiente, en donde los gradientes son calculados usando retropropagación. Es un proceso iterativo en el que ocurre un reajuste de pesos y se minimiza el error de la salida respecto a los valores de la salida de referencia. La técnica cuenta sin embargo con algunas desventajas, siendo la más notoria que el algoritmo de gradiente descendente puede converger a mínimos locales (es decir, un mínimo en la función que relaciona los pesos con el error), en lugar de un mínimo global (el valor óptimo). Si bien este problema puede ocurrir bajo diversas circunstancias, el hecho de que los valores de los pesos definidos por defecto por el algoritmo suelan ser pequeños y aleatorios, hace que sea una problemática inherente a la técnica misma. Por lo tanto, inicializaciones aleatorias de pesos pueden producir diferencias notables en la precisión del modelo.

Implementación elegida: Keras Sequential Model

Si bien la biblioteca scikit-learn cuenta con un clase `MLPRegressor`, que implementa este tipo de redes neuronales, luego de realizar algunas pruebas con esta herramienta exploramos otros desarrollos más eficientes. Finalmente, luego de un estudio de la literatura, terminamos utilizando la biblioteca Keras[85], que es un wrapper de Python para la plaforma de DeepLearning de Google, Tensorflow[86].

Selección de métricas

Como mencionamos anteriormente, la regresión es un problema de minimización de errores. Si bien todas las métricas mencionadas en el Apéndice A, sección A.3.3, pueden funcionar para el problema de regresión, usualmente se utiliza **error cuadrático medio** (ECM o, por sus siglas en inglés, MSE *Mean Squared Error*) como función de pérdida. Se define como el promedio de la sumatoria de los errores cuadráticos y suele ser la más sencilla de calcular y de interpretar. Debido a estas razones la elegimos para esta etapa de nuestro trabajo.

Hiperparámetros del predictor

En esta sección detallaremos el modelo de red neuronal que desarrollamos para nuestro problema de regresión. Keras nos permite crear modelos y evaluarlos usando scikit-learn mediante wrappers provistos por la biblioteca. Los wrappers de Keras requieren una función como argumento de entrada, la cual es responsable de crear el modelo de red neuronal a evaluar. Esto es muy práctico, dado que scikit-learn se destaca particularmente en la evaluación de modelos y, por lo tanto, nos permitirá definir los esquemas de preparación de datos y evaluación de modelos con muy pocas líneas de código.
En relación a los hiperparámetros que pueden configurarse para un modelo de red neuronal, los de mayor relevancia se listan a continuación[l]:

- `neurons`: el número de neuronas en cada capa oculta.
- `weight_constraint`: permite mantener acotado el valor de los pesos que inciden en cada neurona.
- `init_mode`: inicialización de los pesos (ej: distribución uniforme, gaussiana, todos ceros, etc.).
- `activation_function`: función (usualmente no-lineal) que controla cuándo debe "disparar" la neurona o, si la salida son valores continuos, en qué proporción.
- `optimizer`: algoritmo usado para entrenar la red a través de la optimización de una función objetivo.
- `learning_rate`: controla en cuánto actualizar el peso de los nodos al final de cada lote.
- `batch_size` o "tamaño del lote": en el método de descenso por gradiente iterativo es el número de observaciones que se muestran a la red antes de actualizar los pesos.
- `epochs`: número de veces que el conjunto de datos de entrenamiento se muestra a la red durante el entrenamiento.
- `dropout_rate`: proporción de nodos a ignorar en cada capa oculta de la red.

Utilizamos el algoritmo de optimización ADAM[87] debido a que ha demostrado tener una buena performance en una amplia gama de problemas[88]. El algoritmo busca optimizar una función de pérdida de error cuadrático medio, que justamente será la misma métrica que usaremos para

[l] En el Apéndice A, sección A.3.5 se describen las particularidades de algunos hiperparámetros provistos por la herramienta.

evaluar el desempeño del modelo.
Otros hiperparámetros, de igual influencia a los mencionados previamente, son los correspondientes a la estructura de la red en sí. Entre ellos, podemos mencionar la cantidad de capas y la cantidad de neuronas de cada capa de la red (configurables mediante el parámetro `hidden_layer_extension`, que incorporamos para tal fin[m].). En este sentido evaluamos topologías crecientes en "profundidad" y otras crecientes en "amplitud".

Evaluación de la red neuronal en "profundidad": Un análisis que exploramos para evaluar el desempeño de la red neuronal fue agregarle más capas ocultas. Este incremento podría permitir que el modelo extraiga y recombine características de orden superior incluidas en los datos. Para agregar una capa oculta más se define una nueva función que, a partir de nuestro modelo de referencia anterior, genera uno más profundo. Simplemente, al mismo se le debe insertar una nueva línea de código luego de la definición de la primera capa oculta (ver figura 3.16 (b)). Escalamos este procedimiento parametrizando la inserción de la capa mediante un bucle, usando el número de capas indicado en una instancia de `hidden_layer_extension`. Al igual que en el ejemplo de la figura, decidimos que todas las capas ocultas tengan la misma cantidad de neuronas.

Evaluación de la red neuronal en "amplitud": Otro estudio que exploramos, con el objetivo de aumentar la capacidad de representación del modelo, fue el de ampliar el tamaño de la red. Evaluamos el efecto de mantener una capa oculta, y a la vez, ir incrementando la cantidad de neuronas (ver figura 3.16 (c)).

[m] Este hiperparámetro consiste en una tupla de números enteros positivos que denotan, por un lado, la cantidad de capas ocultas y, por otro, un factor de multiplicación de la cantidad de neuronas indicada en el hiperparámetro `neurons`. Por ejemplo, con el valor (3,5) y `neurons`= 13, la red neuronal tendrá 3 capas ocultas de 65 neuronas cada una

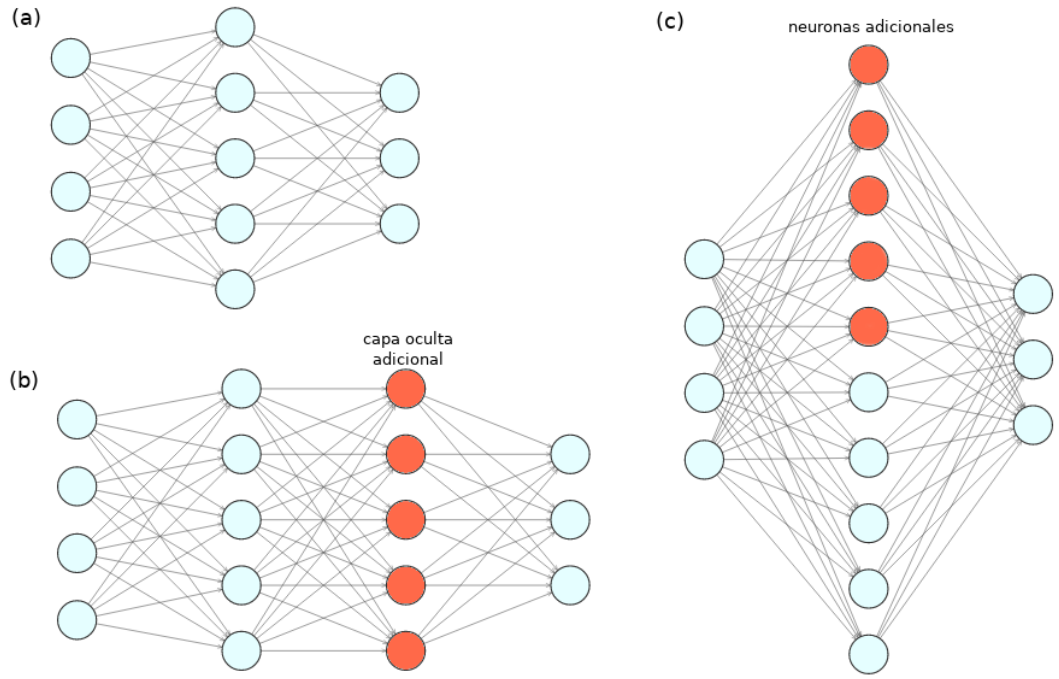


Figure 3.16: a) Red neuronal de referencia con una sola capa oculta. b) Red neuronal obtenida agregando una capa oculta adicional. c) Red neuronal obtenida agregando más neuronas a la capa oculta.

Ajuste del modelo usando Grid Search

Como mencionamos anteriormente, el algoritmo de *Grid Search* selecciona sistemáticamente el valor de cada uno de los hiperparámetros dentro de una lista de opciones previamente definidas, hasta agotar todas las combinaciones posibles, y usa dichos valores para entrenar una instancia del modelo. En una primera etapa, comenzamos definiendo, de forma exploratoria, entre 3 y 10 opciones por cada uno de los 9 hiperparámetros y más de 6 arquitecturas de red, lo que ocasionaba una explosión combinatoria que pronto nos obligó a repensar la manera de definir los valores de forma más selectiva.
Con ayuda de algunas heurísticas[76], limitamos los rangos de cada hiperparámetro a las siguientes opciones:

- `batch_size: 40` .
- `epochs: 50`.
- `optimizer: ADAM`.
- `learn_rate: [0.001, 0.0001]`, valor por defecto del paper de ADAM[87]. Adicionalmente agregamos un valor de un orden de magnitud menor.
- `init_mode: 'lecun_uniform'`, pesos generados en base a una distribución uniforme definida en Lecun et al.[76]

- `activation: tanh`, por estabilidad numérica[76] y porque el rango coincide con el rango de los datos escalados.
- `dropout_rate: 0.2`, valor sugerido en el paper de dropout[89][n].
- `weight_constraint: 1`.
- `neurons: 13`, promedio entre 16 (features de entrada, correspondientes a los pockets) y 10 (features de salida, correspondientes al ligando ideal).
- `hidden_layer_extension: [(1,1), (5,1), (10,1), (15,1),(1,5), (1,10), (1,15), (15,15)]`, correspondiente a 4 arquitecturas de red con 1, 5, 10 y 15 capas ocultas (13 neuronas cada una), 3 arquitecturas de 1 capa oculta con 65, 130 y 195 neuronas, y una arquitectura de red de 15 capas ocultas con 195 neuronas cada una.

Para explorar todos estos valores de hiperparámetro, se debe realizar un total de 16 combinaciones (2 posibles valores correspondientes a `learn_rate` y 8 arquitecturas diferentes).

Resultados del entrenamiento

Por medio de la clase de scikit-learn `GridSearchCV`, realizamos el entrenamiento y validación 3-fold de 16 modelos, generados a partir de las distintas combinaciones de valores de hiperparámetros. La función de pérdida utilizada fue el Error Cuadrático Medio (MSE). Al igual que con k-NN, la misma está expresada como puntaje para poder ser maximizada, de modo que 0 será el mejor puntaje. Valores menores que 0 indicarán un peor rendimiento. Estos pasos fueron aplicados sobre la partición de entrenamiento de cada una de las variantes del dataset:

- Dataset sesgado, un escalado.
- Dataset sesgado, dos escalados.
- Dataset 8L, un escalado.
- Dataset 8L, dos escalados.

Dado que seleccionamos un único valor para la mayoría de los hiperparámetros (es decir, son comunes a todos los modelos), podemos agrupar los resultados en: modelos entrenados con `learn_rate`=0,001 y entrenados con `learn_rate`=0,0001, y comparar distintas variaciones de la arquitectura de la red. Como se verá a continuación, en la mayoría de los gráficos, el valor de MSE varía dentro de un rango muy pequeño, con diferentes valores de mínimo y máximo entre ellas. Dichos valores extremos conforman un rango de un orden de magnitud mayor que los rangos de cada gráfico individual. Con el fin de poder apreciar con mejor detalle la sutiles variaciones de dicha métrica con diferentes topologías de red, utilizaremos escalas independientes (en el eje Y) entre los distintos gráficos.
En primer lugar, analizamos cómo varia el MSE a medida que incrementamos la cantidad de capas ocultas. Como podemos observar en las figuras 3.17 a 3.20 (`learn_rate`=0,001) y 3.21 a 3.24 (`learn_rate`=0,0001), para una cantidad de neuronas fija, el MSE promedio del *3-fold* (líneas) tiende a empeorar con el agregado de más capas. Entre los provenientes del dataset 8L,

[n] En el paper original, está expresado como la probabilidad de conservar cada neurona (1 - `dropout_rate`) de la capa.

en el segundo grupo (figuras 3.23 y 3.24) resulta más evidente la tendencia descendente, es decir, el puntaje empeora. Estos resultados se condicen con la dificultad de entrenar redes neuronales profundas que menciona la literatura del área (usualmente debido a que los gradientes tienden a 0 cuanto mayor es la cantidad de capas que debe atravesar el algoritmo de retropropagación) [90]. En líneas generales, no hubo diferencias sustanciales en el rango de valores de MSE correspondientes a los mismos dataset y escalado (variando `learn_rate`). El MSE de todo el conjunto de gráficos estuvo entre los valores -0,344 y -0,408, considerando el desvío estándar (franjas coloreadas).

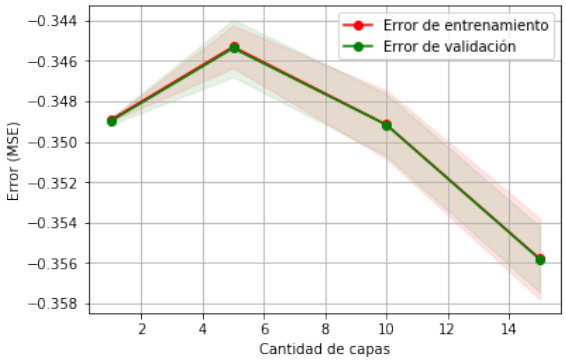


Figure 3.17: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,001), con dataset sesgado y un solo escalado.

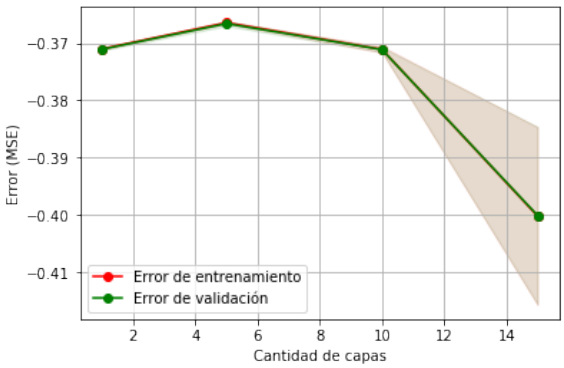


Figure 3.18: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,001), con dataset sesgado y dos escalados.

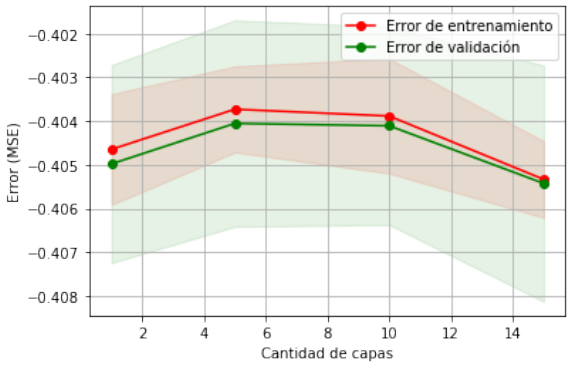


Figure 3.19: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,001), con dataset 8L y un solo escalado.

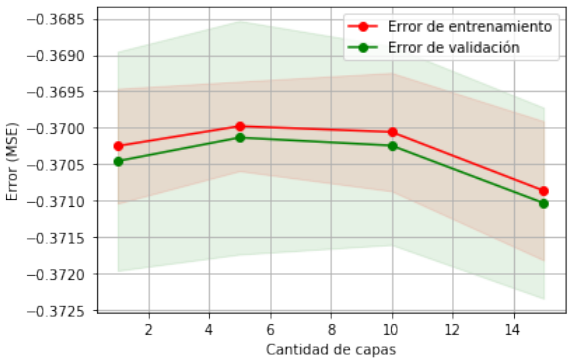


Figure 3.20: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,001), con dataset 8L y dos escalados.

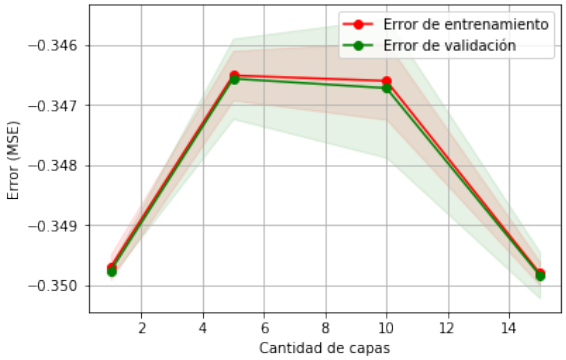


Figure 3.21: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,0001), con dataset sesgado y un solo escalado.

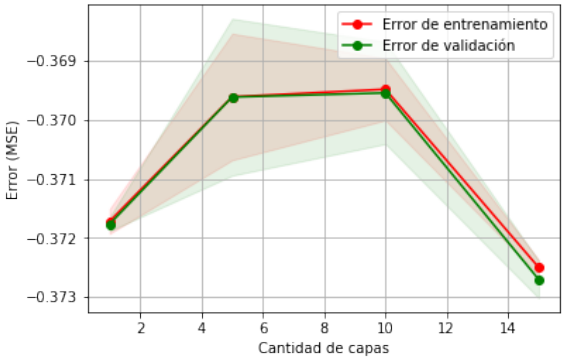


Figure 3.22: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,0001), con dataset sesgado y dos escalados.

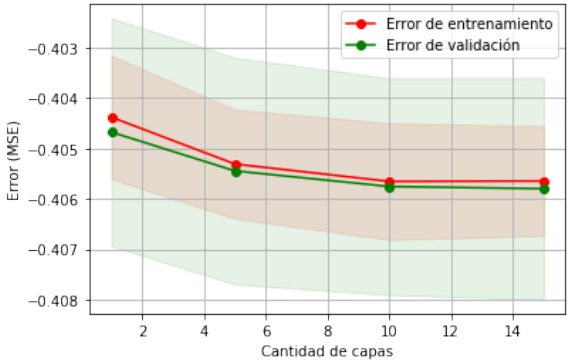


Figure 3.23: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,0001), con dataset 8L y un solo escalado.

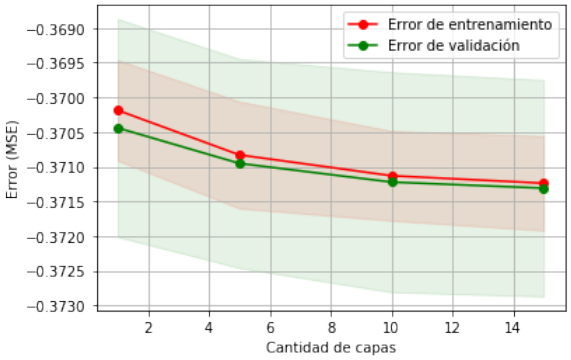


Figure 3.24: MSE promedio y desvío estándar en función de la cantidad de capas ocultas (13 neuronas y learn_rate=0,0001), con dataset 8L y dos escalados.

En segundo lugar, analizamos cómo varía el MSE a medida que incrementamos la cantidad de neuronas de una única capa oculta, nuevamente agrupando los resultados por `learn_rate` (figuras 3.25 a 3.28 con valor de `learn_rate`=0,001 y figuras 3.29 a 3.32, con valor de `learn_rate`=0,0001). A diferencia de lo que ocurre con la evaluación de la red neuronal en profundidad, este conjunto de gráficos exhibe una mejoría general del valor de MSE con el incremento del número de neuronas. Con `learn_rate`=0,001 aún existen descensos de la curva (como el que se logra apreciar en la figura 3.26). Pero para redes entrenadas con un valor más bajo de este parámetro (0,0001), el MSE mostró una tendencia monótona creciente en relación a la cantidad de neuronas.
El valor de MSE en este conjunto osciló entre -0,344 y -0.408, como el rango obtenido en la evaluación de la red en profundidad.

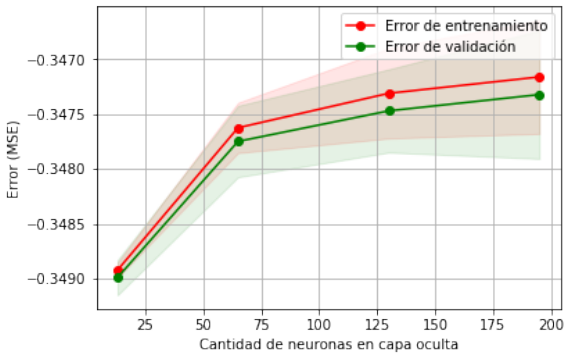


Figure 3.25: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,001), con dataset sesgado y un escalado.

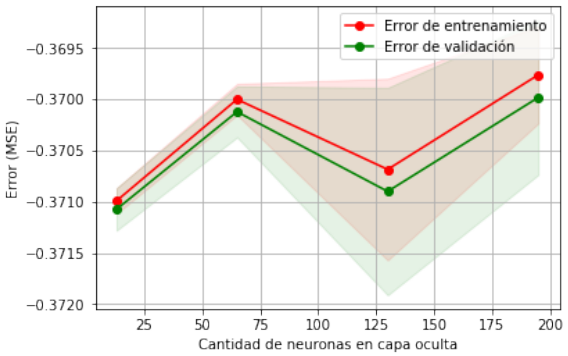


Figure 3.26: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,001), con dataset sesgado y dos escalados.

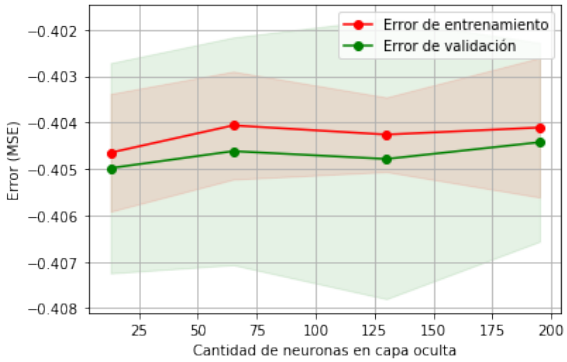


Figure 3.27: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,001), con dataset 8L y un escalado.

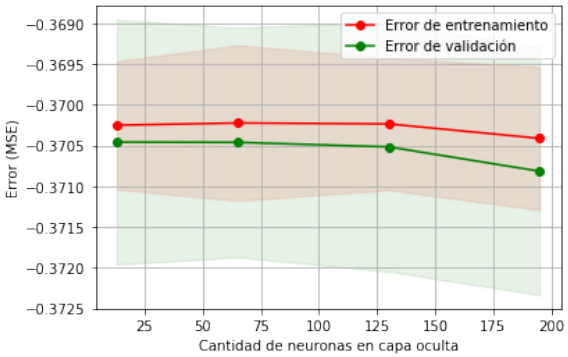


Figure 3.28: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,001), con dataset 8L y dos escalados.

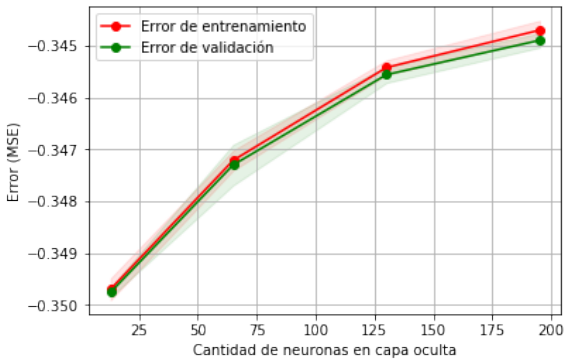


Figure 3.29: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,0001), con dataset sesgado y un escalado.

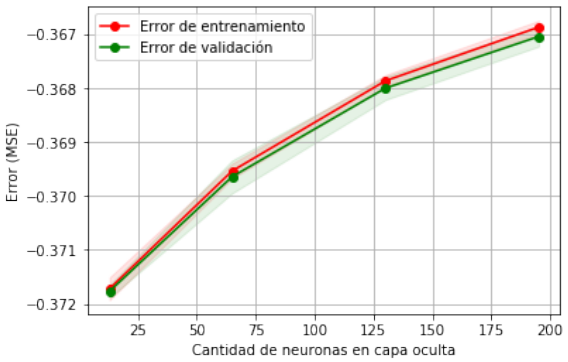


Figure 3.30: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,0001), con dataset sesgado y dos escalados.

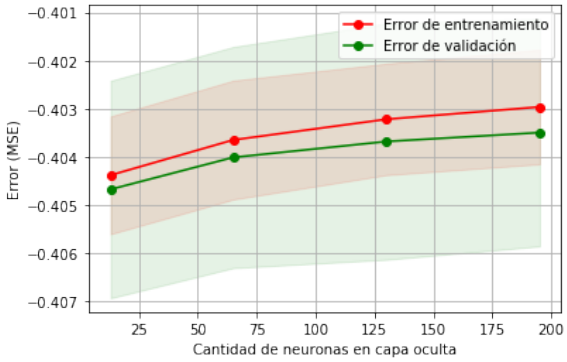


Figure 3.31: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,0001), con dataset 8L y un escalado.

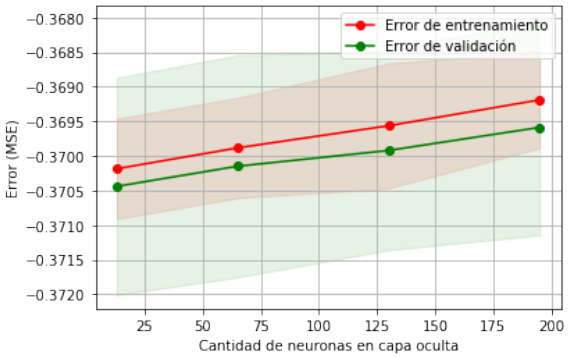


Figure 3.32: MSE promedio y desvío estándar en función de la cantidad de neuronas en única capa oculta (learn_rate=0,0001), con dataset 8L y dos escalados.

Los resultados anteriores parecen sugerir que un incremento en la cantidad de neuronas de una capa ofrece siempre una mejora en el valor de MSE (manteniendo constante todos los demás hiperparámetros), en particular si el valor de `learn_rate` es muy bajo. Mientras que una red neuronal más profunda tiende a obtener valores de MSE cada vez más deficientes.
Sin embargo, la diferencia relativa entre el MSE más alto y el más bajo es muy pequeña, con lo que podría interpretarse como que la combinación de hiperparámetros elegidos no influye radicalmente en la capacidad de aprendizaje de la red neuronal. Posiblemente estamos frente a una limitación intrínseca de los datos actuales ("no hay suficiente información para extraer de los datos").

En cuanto al mejor modelo obtenido por el algoritmo de Grid Search, este encontró que los hiperparámetros, para todos los datasets y escalados, eran los siguientes:

```
{'activation': 'tanh',
 'batch_size': 40,
 'dropout_rate': 0.2,
 'epochs': 50,
 'hidden_layer_extension': (15, 15),
 'init_mode': 'lecun_uniform',
 'input_size': 16,
 'learn_rate': 0.0001,
 'neurons': 13,
 'optimizer': 'adam',
 'output_size': 10,
 'weight_constraint': 1}
```

En otras palabras, una red neuronal profunda (15 capas) y con el valor máximo preestablecido de neuronas (195), entrenada con el valor de `learn_rate` más bajo, 0,0001. El valor de MSE correspondiente a cada dataset con el mejor modelo fue -0.331 con el "Dataset sesgado, un escalado", -0.352 con el "Dataset sesgado, dos escalados", -0.403 con el "Dataset 8L, un escalado" y -0.367 con el "Dataset 8L, dos escalados". Es decir, el valor de MSE de los mejores modelos estuvo en apróximadamente el mismo rango que el de los análisis anteriores.

Feature Importance

Otra opción interesante, además de considerar las distintas configuraciones de hiperparámtros, es la de prestar atención a las features que estamos utilizando. Existe una técnica, denominada *features importance*, que busca determinar cuáles son las features mas relevantes partiendo de un modelo de predicción. En esta técnica se suelen utilizar estimadores (típicamente **Random Forest**) para probar escenarios predictivos y sacar conclusiones *a posteriori*. Su uso en `sci-kit learn` es bastante simple, dado que el mismo modelo de Random Forest, luego de ser entrenado, guarda este valor a modo de lista. Generamos dichos valores para los dos dataset que usamos en esta etapa: 8L y sesgado. Los resultados pueden verse en la tabla 3.1.

Feature de pocket	**Importancia**	
	Dataset 8L	**Dataset sesgado**
mean_local_hydrophobic_density	0.8915687500560537	0.0713157173928983
apolar_alpha_sphere_proportion	0.058755512038485856	0.0
sasa_apolar	0.03817308514187147	0.028593821840447564
proportion_polar_atoms	0.01150265276358898	0.0
number_spheres	0.0	0.0
sasa_total	0.0	0.0
sasa_polar	0.0	0.0
mean_alpha_sphere_radius	0.0	0.0
mean_alpha_sphere_solvent_access	0.0	0.0
charge_score	0.0	0.0
alpha_sphere_density	0.0	0.0
com_alphasphere_distance	0.0	0.0
flexibility	0.0	0.0
lr_pocket	0.0	0.0
mr_pocket	0.0	0.9000904607666541
sr_pocket	0.0	0.0

Table 3.1: Features de los pockets junto a sus respectivos valores de importancia. En la columna central se listan los valores para el dataset 8, y en la columna de la derecha, los valores correspondientes al dataset sesgado.

Como se puede ver en la tabla 3.1, en ambos casos las propiedades referentes a la polaridad parecen tener una gran relevancia[o]. En el caso del dataset completo, con mayor valor aparece el radio medio de los pockets, lo cual dice que, en este caso, la forma se vuelve muy significativa. Presumiblemente esta diferencia con respecto al dataset 8L tenga que ver con aquellos pockets con muchos acoplamientos, los cuales quizás posean características especiales en tamaño que los hacen muy atractivos.
En función de estos resultados hicimos corridas de Grid Search entrenando redes neurnales con configuraciones de hiperparámetros similares a las ya utilizadas en las pruebas anteriores, con-

[o]Cabe aclarar que la técnica de *features importance* actúa sobre el conjunto de entrada pero no sobre el de salida, es decir, no brinda información acerca de qué features de la salida gozan de una buena predictibilidad. Realizar este trabajo es una ardua tarea manual y no fue llevada a cabo en este trabajo de tesis. En el capítulo 8 (Trabajo Futuro) se describen algunas reflexiones al respecto de este tema.

siderando únicamente el dataset 8L a partir de 2 subconjuntos de features: el correspondiente a los primeros cuatro features de la tabla 3.1 y los relativos a la primera componente de PCA (ver sección 2.2.5, capítulo 2), que refería a la información volumétrica del pocket. La hipótesis detrás de esta última opción tiene que ver con validar si, más allá de los resultados de aplicar *feature importance*, la información volumétrica tiene un buen impacto en la predicción. Las configuraciones pueden verse en la tabla 3.2.

Hiperparámetros	**Valores**
activation	tanh
batch_size	40
dropout_rate	0,0
epochs	10, 50
hidden_layer_extension	(variable)
init_mode	lecun_uniform
input_size	10 (features asociados a la primer componente de PCA) 4 (features más importantes en base a feature importance)
learn_rate	0,001, 0,0001
neurons	13
optimizer	ADAM
output_size	10
weight_constraint	2

Table 3.2: Valores hiperparámetros utilizados durante la etapa de Grid Search.

Para las propiedades seleccionadas por *feature importance*, el input size es igual a 4^p. La configuración con mejor resultado utiliza `hidden_layer_extension` es igual a `(15, 15)`, y el `learn_rate` igual a 0.0001. Luego de generar el modelo, obtuvimos un puntaje de -0.3715, lo cual no mejora los puntajes anteriores.
En el caso en el que tomamos las features asociadas a la primer componente de PCA, el input size es igual a 10 y la configuración con mejor resultado utilizó `hidden_layer_extension` igual a `(2, 20)`, y `learn_rate` igual a 0.0001. En este caso, el puntaje obtenido fue de-0.3687, similar al caso anterior.
Lamentablemente, los valores de error no fueron muy distintos a los obtenidos con los modelos que utilizan el total de las features de entrada, de forma que decidimos descartar estas variantes.

3.4 Obtención de los ligandos *reales*

Una vez finalizado el entrenamiento del modelo de regresión, con el vector de features de un pocket como entrada del modelo, predecimos el vector de features del ligando ideal. En la siguiente etapa, procedemos a buscar entre el total de ligandos de nuestro dataset (archivo `ligands.csv`) los n ligandos más similares al ligando ideal, para un valor de n preestablecido

[p] En toda esta etapa se modificaron los modelos de scaling para que tome como parámetro esas 4 ó 10 features dependiendo del caso, de la misma forma se tuvieron que recalcular las particiones de train/test.

(ver figura 3.3, etapa B de nuestro pipeline).
Para ello primero clasificamos los ligandos mediante técnicas de clustering. Luego, utilizando el mismo modelo de clasificación, predecimos la clase del ligando ideal.

3.4.1 Clustering (K-means) y selección de ligandos similares

Decidimos utilizar la técnica K-means para realizar el clustering de los ligandos, cuyas features fueron previamente escaladas como se indica en la sección 3.2.1. De acuerdo a los resultados de las figuras 3.1 y 3.2, cualquier valor de k superior a 200 nos asegura una inercia baja, así que optamos por aquel que nos proporcione un tamaño de cluster moderado: con $k = 800$ obtenemos en promedio 200 ligandos por cluster.
A continuación, para seleccionar los n ligandos que devolverá el sistema de recomendación elegimos aquellos más cercanos al ligando ideal (en distancia euclideana) que pertenecen al mismo cluster que este último.

3.5 Evaluación de las recomendaciones

Para evaluar la calidad de las recomendaciones del pipeline, hicimos uso de las muestras de control que habíamos separado previamente al entrenamiento de los modelos de regresión (ver sección 3.2.2). El primer grupo de control lo usamos para comparar la performance entre pipelines con la siguiente configuración de red neuronal para todos los casos:

```
{'activation': 'tanh',
 'batch_size': 40,
 'dropout_rate': 0.2,
 'epochs': 50,
 'hidden_layer_extension': (15, 15),
 'init_mode': 'lecun_uniform',
 'input_size': 16,
 'learn_rate': 0.0001,
 'neurons': 13,
 'optimizer': 'adam',
 'output_size': 10,
 'weight_constraint': 1}
```

Luego, utilizando el pipeline con el que obtuvimos mejores resultados, repetimos el análisis con el segundo grupo de control.

En cuanto a las métricas de performance, optamos por utilizar:

- **Métrica 1:** la exactitud (en inglés, *accuracy*) de la predicción del cluster al cual pertenece el ligando ideal de un pocket, en relación a cada uno de los ligandos "reales" de este último (ver figura 3.33). En esta métrica, un valor de 1 indicaría que todos los ligandos ideales se encuentran asociados al mismo cluster que sus ligandos reales. Un valor de 0 se obtendría en el caso de que ninguno de los ligandos ideales se encuentren asociados al cluster de sus ligandos reales.

- **Métrica 2:** La proporción de pockets del conjunto de control para los cuales existe al menos un ligando real en el mismo cluster que su ligando ideal (ver figura 3.33). Un valor de 100% indicaría que todos los ligandos ideales se encuentran asociados al mismo cluster que alguno de sus ligandos reales. Un valor de 0% se obtendría en el caso de ninguno de los ligandos ideales se encuentren asociados al cluster de sus ligandos reales.
- El área bajo la curva ROC (o ROC AUC), detallada más adelante.

Evaluación de rendimiento utilizando la métrica 1

La exactitud se define como la proporción de valores predichos correctamente sobre el total de casos examinados. En particular, nos interesaba conocer qué proporción de los pares pocket-ligando del primer conjunto de test tenían coincidencia entre el cluster al cual pertenecía cada ligando real de un pocket y el cluster determinado para su ligando ideal. Para ello, primero predecimos el ligando ideal de un pocket del conjunto. Luego, utilizando el modelo de clustering que generamos previamente, determinamos a qué cluster (identificado por un `cluster_id`) pertenecían, por un lado, los ligandos reales y, por otro, el ligando ideal. Finalmente, calculamos la cantidad de coincidencias que había entre el `cluster_id` del ligando ideal y el `cluster_id` de los correspondientes ligandos reales. Repetimos el procedimiento con los demás pockets. El valor de *accuracy* lo calculamos dividiendo la cantidad global de aciertos sobre el total de acoplamientos del conjunto de control. Los resultados obtenidos aplicando la métrica 1 se pueden observar en la tabla 3.3.

Dataset	Métrica 1 (Porcentaje)
Dataset sesgado, un escalado	0.001760 (0,18%)
Dataset sesgado, dos escalados	0.001853 (0,19%)
Dataset 8L, un escalado	0.001817 (0,18%)
Dataset 8L, dos escalados	0.028931 (2,89%)

Table 3.3: Valores de accuracy (métrica 1), en la predicción de cluster de pertenencia, obtenidos para cada combinación de dataset y escalado

Nuestros resultados, aplicando la métrica 1, indican valores muy bajos de coincidencia entre ligando ideal y ligando real. El mejor valor lo obtuvo el Dataset 8L con dos escalados (16 veces más aciertos que los demás). La explicación que encontramos a esta situación es que, para obtener valores máximos, el ligando ideal de un pocket sólo puede pertenecer a un único cluster. De modo que, si el modelo de clustering no consigue agrupar ligandos reales del pocket en una misma clase, aún en el mejor caso no llegaremos a valores del 100% de exactitud. Por ello, consideramos una métrica menos restrictiva: la métrica 2.

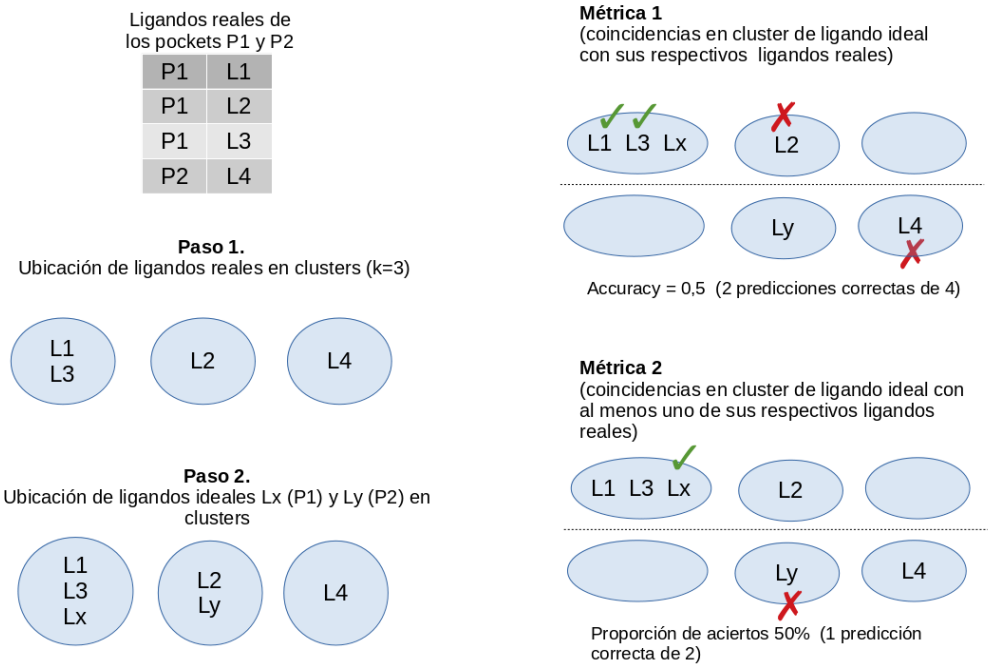


Figure 3.33: Descripción gráfica de la evaluación en un conjunto de test de ejemplo que contiene dos pockets y sus respectivos acoplamientos. Luego de determinar el cluster al cual pertenece cada ligando real (paso 1) y cada ligando ideal (paso 2), realizamos la medición de la tasa de coincidencias utilizando dos criterios diferentes (métricas 1 y 2).

Evaluación de rendimiento utilizando la métrica 2

Al aplicar la métrica 2 a los datos de nuestro dataset control, obtuvimos los resultados que se muestran en la tabla 3.4. Si bien los valores obtenidos fueron más altos (con respecto a los obtenidos con la métrica 1, ver tabla 3.3), la diferencia fue marginal. Nuestra hipótesis, con respecto a estas pequeñas diferencias, es que existe un problema de fondo en la etapa de ubicación del ligando ideal en un cluster y, potencialmente, en la etapa de predicción de dicho ligando. De esta manera, resulta válido pensar que la proporción de coincidencias mejore significativamente si eligiésemos un valor de k pequeño para el clustering de ligandos. Este planteo tiene un punto débil: al hacerlo, se pierde inevitablemente cierta especificidad de los resultados, comprometiendo la calidad general de las predicciones brindadas. En un caso extremo, si los ligandos estuviesen agrupados entre dos clusters, con la métrica 2 podríamos obtener una tasa de aciertos sustancialmente mayor a la actual pero, a su vez, aumentaría considerablemente la proporción de ligandos irrelevantes para el pocket objetivo.

Dataset	Métrica 2
Dataset sesgado, un escalado	0.27%
Dataset sesgado, dos escalados	0.25%
Dataset 8L, un escalado	0.19%
Dataset 8L, dos escalados	2.95%

Table 3.4: Porcentaje de coincidencias entre el cluster del ligando ideal de cada pocket con al menos uno de los clusters de los ligandos reales del pocket, obtenidos para cada combinación de dataset y escalado

A pesar de la baja proporción de aciertos a nivel global (definidos por la pertenencia al mismo cluster), nos propusimos continuar la evaluación del rendimiento del sistema, pero a nivel intra-cluster. Esta vez, indagando más a fondo aquellos casos en los existía al menos un ligando real en el cluster que contenía al ligando ideal.

La siguiente métrica trabaja con los conceptos de verdadero positivo y verdadero negativo (predicciones correctas), y de falso positivo y falso negativo (predicciones incorrectas), por lo que necesitamos definir previamente cada uno de éstos para hacer uso apropiado de dicha métrica. En el contexto de evaluación de una recomendación de ligandos correspondientes a un pocket, definimos los ligandos que interactúan con dicho pocket en el conjunto de test como una clase "positiva", y cualquier otro como "negativa". Si este fue incluido en el listado recomendado, hablamos de un verdadero positivo; en caso contrario, se trata de un falso negativo. Entre las clases negativas, los compuestos incluidos en el listado de recomendación son falsos positivos, pero si están fuera del listado, serán verdaderos negativos.

Evaluación de rendimiento: curva ROC

Para graficar qué tan bien funciona el sistema de recomendación recurrimos a la técnica de **curva ROC**. Si bien tiene su origen en la teoría de detección de señales, la curva ROC (acrónimo de *Receiver Operating Characteristic* o, en español, Característica Operativa del Receptor) puede utilizarse para medir la razón (ratio) de verdaderos positivos frente a la de falsos positivos según varía el umbral de discriminación[q]. En el caso de la recomendación de ligandos, consideramos como dicho umbral al tamaño del conjunto recomendado[91]; comenzamos con un solo ligando (el más cercano al ligando ideal), y luego incorporamos sucesivamente los siguientes por orden de distancia hasta abarcar la totalidad de ligandos del cluster.

[q] Usualmente proviene de un intervalo continuo, pero también pueden usarse valores ordinales si se cuenta con la cantidad considerable.

Para cada valor del umbral, medimos la razón de verdaderos positivos (VPR) y la razón de falsos positivos (FPR), que se definen como:

$$VPR = \frac{VP}{VP + FN}$$

$$FPR = \frac{FP}{FP + VN}$$

Con estas dos medidas se obtiene un punto de la curva ROC.
El área total bajo la curva (abreviado en inglés AUC, *Area Under Curve*) proporciona una medida de performance del clasificador. Cuanto más cercano a 1 sea el valor de AUC, mejor es el desempeño general del modelo. En cuanto a la curva, se busca que ésta se acerque lo más posible a la esquina superior izquierda del gráfico de ROC (i.e., curva ROC del clasificador perfecto). Un valor de ROC AUC cercano a 0.5 y una curva ROC similar a la función identidad, se puede interpretar como que el modelo tiene el mismo desempeño que una selección aleatoria. Para entender mejor el concepto de curva ROC, mostramos en la figura 3.34 qué es un buen y un mal resultado en función de los ratios mencionados anteriormente.

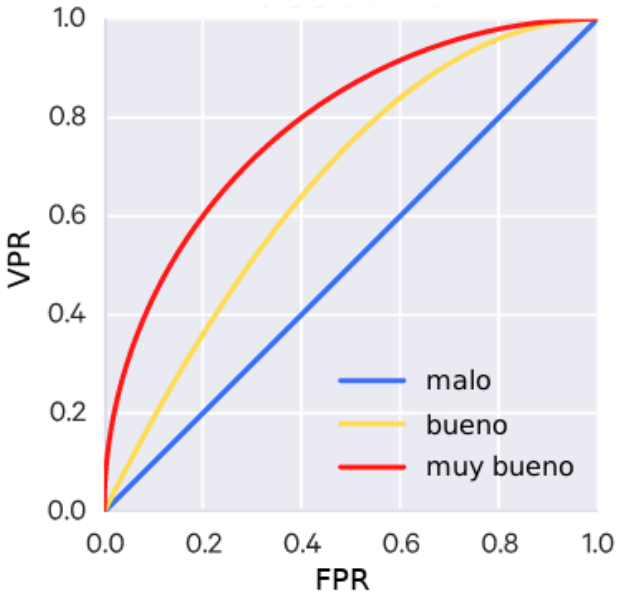


Figure 3.34: Curvas ROC

Otra forma de interpretar VPR y FPR es la siguiente:

- VP son los positivos clasificados correctamente, y VN son los "positivos que quedaron fuera" (es decir, negativos incorrectamente clasificados). Se busca un valor alto de VPR, porque sugiere que incluimos los positivos que había que incluir. Sin embargo, sólo con esto no alcanza, hay que ver la contraparte: los casos negativos.

- FP son los positivos mal clasificados como positivos (es decir, negativos incluidos por error en los positivos), y VN son los negativos correctamente clasificados. Se busca un valor bajo de FPR, porque significa que se están incluyendo pocos negativos incorrectamente.

Resultados

Para cada uno de los pockets del conjunto de test con los que obtuvimos predicciones correctas en sus ligandos ideales según la métrica 2 (**"subconjunto ROC M2"**), graficamos la curva ROC y calculamos el ROC AUC. En primer lugar, obtuvimos los valores de VPR y FPR para cada valor del umbral haciendo uso del método `roc_curve` de scikit-learn. Éste toma dos parámetros: una lista de etiquetas con las predicciones positivas y negativas (por defecto, 1 y 0, respectivamente), y una lista de puntajes. Este último sirve, por un lado, para establecer un orden en las predicciones del primer parámetro (a modo de ranking, cuanto mayor es el valor, más cerca de la primera posición estará la predicción) y, por otro, para agrupar predicciones con

el mismo puntaje y determinar una probabilidad de la clase positiva. En nuestro caso, utilizamos la medida $\frac{1}{d_i}$, donde d_i es la distancia del i-ésimo compuesto más lejano al ligando ideal, para ordenar los compuestos del cluster dentro del ranking (posicionando los más cercanos en el tope de la lista, ya que el ranking se encuentra ordenado de mayor a menor).
Con los valores de VPR y FPR graficamos las curva ROC, y obtuvimos el área bajo la misma con el método `auc`. En la tabla 3.5 exponemos el valor de ROC AUC promedio, para cada uno de los pipelines.

Dataset	ROC AUC promedio (desvío estandar)
Dataset sesgado, un escalado	0.52 (± 0.27)
Dataset sesgado, dos escalados	0.40 (± 0.20)
Dataset 8L, un escalado	0.57 (± 0.34)
Dataset 8L, dos escalados	0.35 (± 0.27)

Table 3.5: ROC AUC promedio para las recomendaciones de ligandos de cada uno de los pockets pertenecientes a la muestra de control.

En todos los casos, el valor promedio de ROC AUC oscilaba alrededor de 0.5, en los datasets con un solo escalado, lo que nos indicaba que el recomendador no funcionaría mejor que seleccionar los valores al azar[r]. Sin embargo, el desvío estandar nos sugería la existencia de casos para los que el recomendador tenía buen desempeño, por lo que analizamos además con qué proporción de los pockets del "subconjunto ROC M2" se obtenía ROC AUC mayores que 0.6, 0.7, 0.8 y 0.9. Los resultados pueden verse en la tabla 3.6. De dicha tabla, podemos destacar que para el "Dataset 8L, un escalado", en más del 60% de los casos, el ROC AUC fue mayor que 0.6, lo que demuestra un mejor desempeño general del recomendador para los pocos casos en los que logra que el ligando ideal y un ligando real queden contenidos en el mismo pocket. Le sigue en performance el "Dataset sesgado, un escalado" (40%).
En cuanto al "Dataset 8L, dos escalados", no tiene un buen desempeño comparado con los dos anteriores (en 21% de los casos consigue ROC AUC $>$ 0,6) con respecto al subconjunto de pockets sobre el que realizamos las mediciones. Sin embargo, si tenemos en cuenta la cantidad total de pockets en el conjunto de test y las proporción de aciertos indicadas en la tabla 3.4, este 21% representa un 0,64% del total de pockets, en comparación a 0,12% del "Dataset 8L, un escalado", 0,11% del "Dataset sesgado, un escalado" y 0,07% del "Dataset sesgado, dos escalados". Incluso, si fuéramos muy conservadores y exigiésemos ROC AUC $>$ 0,9, el 4,35% del subconjunto de pockets equivaldría a 0,13% del total, valor aún mayor que los porcentajes que enumeramos.

[r]Notar que el ROC AUC promedio para los dos datasets con dos escalados es menor que 0.5, es decir, el clasificador binario se desempeña peor que seleccionar valores al azar. Este resultado puede invertirse intercambiando predicciones positivas con negativas para obtener el valor complementario de ROC AUC. Sin embargo, en nuestro esquema sería incorrecto hacerlo, porque implica que tendríamos que considerar como negativos los ligandos reales de cada pocket

Dataset	>0.6	>0.7	> 0.8	> 0.9
Dataset sesgado, un escalado	40,00%	25,00%	15,00%	15,00%
Dataset sesgado, dos escalados	27,78%	5,56%	0,00%	0,00%
Dataset 8L, un escalado	61,54%	38,46%	38,46%	30,77%
Dataset 8L, dos escalados	21,74%	12,56%	7,73%	4,35%

Table 3.6: Proporción de pockets en el "subconjunto ROC" para los cuales ROC AUC es mayor al valores indicados.

En conclusión, en cuanto a desempeño global sobre el primer conjunto de test, **con el "Dataset 8L, dos escalados" se consiguen mejores recomendaciones que con los tres restantes.** Sobre el total de pockets, 0,64% de las veces funciona ligeramente mejor que seleccionar al azar, y el 0,13%, mucho mejor que selección aleatoria. A fines prácticos, esto significa que, si un investigador ingresara las features de un pocket como entrada del modelo, sólo encontraría un ligando novedoso (con buena capacidad de acoplarse al pocket en cuestión) con una probabilidad del 0,13% entre los compuestos recomendados.
El pipeline este último dataset fue el que seleccionamos para evaluar el desempeño con la segunda muestra de control.

Evaluación de rendimiento con segunda muestra de control

Evaluamos el desempeño del pipeline elegido con la segunda muestra de control, utilizando las tres métricas anteriores, y obtuvimos los siguientes resultados:

- métrica 1, para la predección del cluster al cual pertenecían ligando ideal y ligandos reales, fue de 0.032310 $(3,23\%)$.
- métrica 2, correspondiente al porcentaje de pockets de la muestra para los cuales el ligando ideal resultó incluido en el cluster de alguno de los ligandos reales, fue del 3,66% (262 pockets, de un total de 7.166).
- El área bajo la curva ROC, de las recomendaciones para los pockets anteriores, fue en promedio 0,38 ($\pm$ 0,28). De dicho subconjunto, en 28,24% de los casos (1,02% del total) el AUC fue mayor a 0,6; en 15,27% de los casos (0,56% del total) el AUC fue mayor a 0,7; en 9,92% (0,36% del total), fue mayor a 0,8 y **en 3,82% de los casos (0,14% del total), el AUC fue mayor a 0,9.**

De esta forma corroboramos el resultado de las medidas de performance que habíamos obtenido con la primera muestra de control. En síntesis, con la segunda muestra de control, obtuvimos resultados similares a la primera muestra de control.

Discusión

Es importante mencionar que la evaluación fue realizada sobre la recomendación de ligandos que se puedan acoplar a un pocket. No hay que perder de vista que una estructura proteica puede tener varios pockets y, en consecuencia, la cantidad de ligandos que pueden acoplarse a la misma se incrementa. Incorporar información sobre proteínas (por ejemplo, PDB ID o familias de proteínas), en el esquema de evaluación hubiese requerido crear nuevas particiones de train/test con otro criterio de "muestra balanceada", lo cual habría extendido significativamente

el volumen de trabajo de esta etapa. Por esta razón, optamos por mantener el esquema planteado anteriormente.

3.6 Selección final de modelos

Finalmente, los predictores elegidos fueron Redes Neuronales y K-means. Las configuraciones de parámetros elegidas fueron las siguientes:

- Red neuronal:

```
{activation: 'tanh',
 batch_size: 40,
 dropout_rate: 0.2,
 epochs: 50,
 hidden_layer_extension: (15, 15),
 init_mode: 'lecun_uniform',
 input_size: 16,
 learn_rate: 0.0001,
 neurons: 13,
 optimizer: 'adam',
 output_size: 10,
 weight_constraint: 1}
```

- Para el caso de K-means, optamos por un clustering de ligandos con $k = 800$ y los restantes parámetros quedaron con sus valores por defecto.

3.7 Conclusiones

En este capítulo, tuvimos la oportunidad de explorar distintos modelos alternativos para modelar la predicción de pocket→ligandos y desarrollamos argumentos que nos permitieron tomar decisiones respecto a cuáles utilizar. Muchos de los esquemas descartados fueron dejados de lado debido a las características del dataset con el que contábamos, y no por razones vinculadas a las técnicas en sí mismas. En el capítulo 8 (Trabajo Futuro) detallamos posibles implementaciones utilizando estas herramientas a partir de reformulaciones de los datos originales.
Prestando atención a un subconjunto acotado de técnicas, pudimos profundizar en conceptos como el de Grid Search, y las distintas formas de medir los rendimientos de los modelos, así como también los esquemas de validación del pipeline general. Esto fue muy provechoso para poder comprender de manera más amplia el panorama general del área de aprendizaje automático.
La técnica de clustering propuesta, K-Means, constituye un modelo simple que nos permitió resolver la etapa de agrupamiento del pipeline. Debido a que, para proveer un sistema de recomendación con buen rendimiento, decidimos focalizarnos en la etapa de regresión, no profundizamos en diferentes métricas ni en otras técnicas alternativas a K-means. Consideramos que puede ser interesante para trabajos futuros indagar con más detalle en este punto.
Es muy importante destacar el hecho de que, en el pipeline propuesto, la etapa de entrenamiento del modelo de regresión (predicción de ligando ideal) y la etapa de clustering son independientes

entre sí. Esto constituye una desventaja debido a que el regresor no puede retroalimentarse con información proveniente del final del pipeline, referente a la calidad de las recomendaciones. En este sentido, puede ser provechoso profundizar en cómo entrenar ambos modelos en simultáneo como si fuesen uno solo.
Por otro lado, comprendimos que la validación de un sistema de recomendación es una tarea compleja. Las distintas estrategias se encuentran estrechamente vinculadas a las características de los datos y a los criterios que el profesional del área (el usuario final del sistema) considere necesarios. En nuestro caso, debido a la escasez de datos de interacciones pocket-ligando y a la gran presencia de ligandos totales, optamos por un esquema de validación poco restrictivo, que permitiese dar una idea al lector de las bondades de los modelos.
Por último, al igual que los scripts de datos utilizados para la generación y curado de nuestros datasets, dejamos disponibles todo el código de procesamiento de esta etapa para futuros usos. Creemos que más allá de los resultados de este trabajo de tesis, se cumplió otro de los objetivos referido a proveer un framework para el entrenamiento y evaluación de técnicas de aprendizaje automático dentro del ámbito de Virtual Screening.

4 | Incorporación del predictor a LigQ

Una vez obtenido un modelo predictor: *pocket* → *ligandos*, procuramos integrarlo a una herramienta de *virtual screening* ya existente, que actualmente es gerenciada por la Plataforma Bioinformática Argentina (BIA): LigQ[a]. La herramienta, en esencia, sigue los siguientes lineamientos: "*que todos los desarrollos llevados a cabo sean reproducibles y que su grado de automatización sea lo mayor posible, sin que esto afecte la calidad de los resultados obtenidos [...] y garantizar su accesibilidad mediante herramientas online, así como poner a disposición todo lo desarrollado en forma abierta para que cualquier usuario pueda tanto consultar como* ***extender*** *los desarrollos realizados*"[37]. Es principalmente por ello, porque cuenta con acceso público y gratuito, la razón por la cual nos resulta muy importante incorporar nuestro sistema de recomendación a ella, permitiendo así a los investigadores poder obtener las predicciones de nuestros modelos de una manera ya conocida y de fácil uso.

A continuación describiremos LigQ, su arquitectura y las funcionalidades que actualmente provee. Detallaremos todos los pasos que seguimos para extender la funcionalidad provista por éste, desde la actualización del software original hasta la modificación del código del mismo que finalmente permite incorporar nuestros modelos de predicción al pipeline principal.

4.1 LigQ v1.0: Proyecto original

LigQ es una de las múltiples herramienta desarrolladas en el marco de la tesis doctoral de Leandro Radusky[37], que intenta "*aportar información valiosa (...) aplicada a dilucidar causas y plantear soluciones para el tratamiento de enfermedades (de origen bacteriano, genéticas, etc.)*". Dicho trabajo comprende múltiples objetivos, que fueron cubiertos por un vasto conjunto de programas integrados que "*combinan, analizan y generan nuevos datos (en caso de ser necesario), utilizando métodos existentes o desarrollando nuevos métodos computacionales*"[37]. El listado completo de herramientas desarrolladas en dicho trabajo incluye:

TuberQ: asiste en la elección de blancos proteicos en base a su drogabilidad estructural y permite manejar datos a escala genómica (extraer blancos drogables dado el genoma de un organismo).

LigQ: dado un blanco proteico, determina una lista de compuestos candidatos que poseen buenas probabilidades de acoplamiento al mismo, afectando finalmente su función.

VarQ: permite obtener de bases de datos de acceso público las mutaciones reportadas para una proteína de interés, y permite analizar los efectos estructurales que tendrán tanto esas mutaciones como predecir el efecto de otras que resulten de interés en el momento del análisis, y que no se encuentren aún descritas.

[a] La versión original se encuentra actualmente corriendo en un servidor del BIA y es de público acceso (`http://ligq.qb.fcen.uba.ar`)

4.2 Actualización del desarrollo original

Dado que una vez concluído el trabajo de tesis doctoral anteriormente mencionado, el código de LigQ no se continuó actualizando, en una primera instancia nuestro trabajo estuvo orientado a resolver problemas de dependencias. Para ello instalamos y pusimos en funcionamiento una instancia de LigQ en un ambiente nuevo de desarrollo, fuera del servidor productivo. En este proceso agregamos al entorno de ejecución una capa de abstracción y aislamiento del resto del sistema, para contar con una mayor flexibilidad a la hora de introducir cambios. De esta manera sumamos la garantía de la portabilidad del software (es decir, que pueda ser instalado en cualquier ambiente de desarrollo) y asegurar la reproducibilidad de sus resultados.
Adicionalmente, durante los primeros meses, resolvimos una amplia variedad de inconvenientes relacionados a requerimientos no funcionales. En suma, las adaptaciones realizadas fueron:

- *Containerización* de la aplicación: usamos *docker containers*[92] para la capa de virtualización sobre la cual realizamos la instalación de LigQ y componentes asociados. De esta manera se brinda a su vez la posibilidad de ser instalado en una versión moderna de GNU/Linux.
- Alojamiento del proyecto en un repositorio de código[b].
- Migración del código escrito en Python versión 2 a Python versión 3, con el fin de poder trabajar con bibliotecas más modernas y así facilitar desarrollos posteriores[c].
- Instalación de dependencias de LigQ en sus versiones actualizadas: bibliotecas de sistema, paquetes de Python 3 y aplicaciones de terceros (*third-party*).
- Eliminación de referencias a componentes que no forman parte del proyecto actual, pero que se encuentran referenciados dentro del código fuente original de LigQ (e.g. VarQ).
- Escritura de un manual de instalación (que se encuentra en el archivo `README.md` del repositorio).
- Refactoring de métodos para evitar la duplicación de código y optimizar procesos.
- Con el fin de validar el correcto funcionamiento de la nueva herramienta de predicción, se pusieron en funcionamiento una batería de test de unidad (*unit tests*). Dado que el software original no contaba con un framework de testing automatizado, desarrollamos un breve módulo a tal fin, que luego sumamos al sistema global. Debido a esa misma razón, acotamos el alcance de esta etapa únicamente al código modificado, ya que realizar tests *end-to-end* para el sistema entero involucra un trabajo mucho mayor, que excede por demás la temática de esta tesis (para más detalle, consultar chapter 8).

Integración con el resto de las herramientas

Debido a que el trabajo de Leandro Radusky ataca una gama de problemas de forma conjunta, los componentes que forman parte del software definitivo (TuberQ, LigQ y VarQ) no fueron diseñados para trabajar de manera independiente. Esto devino en el hecho de que ciertas

[b] Se encuentra alojado en un `https://git.exactas.uba.ar/pturjanski/LigQ`, un repositorio privado.

[c] Un detallado más extenso de los problemas de compatibilidad entre versiones que fueron resueltos puede encontrarse en el Appendix B.

funcionalidades usualmente contaran con fragmentos intercalados de código correspondientes a varios de ellos[d], haciendo que la cantidad de dependencias de paquetes a resolver fuera bastante más grande de lo estrictamente necesario para nuestros fines. Gracias a la enorme colaboración de Leandro Radusky, fuimos desactivando estos circuitos (ya sea eliminando código o simplemente comentándolo) hasta llegar a un software funcional.

4.3 Extensión de funcionalidad

4.3.1 Arquitectura original

La arquitectura original de LigQ consta de cuatro módulos que pueden ser ejecutados de manera independiente o encadenada, como se muestra en la Figure 4.1.

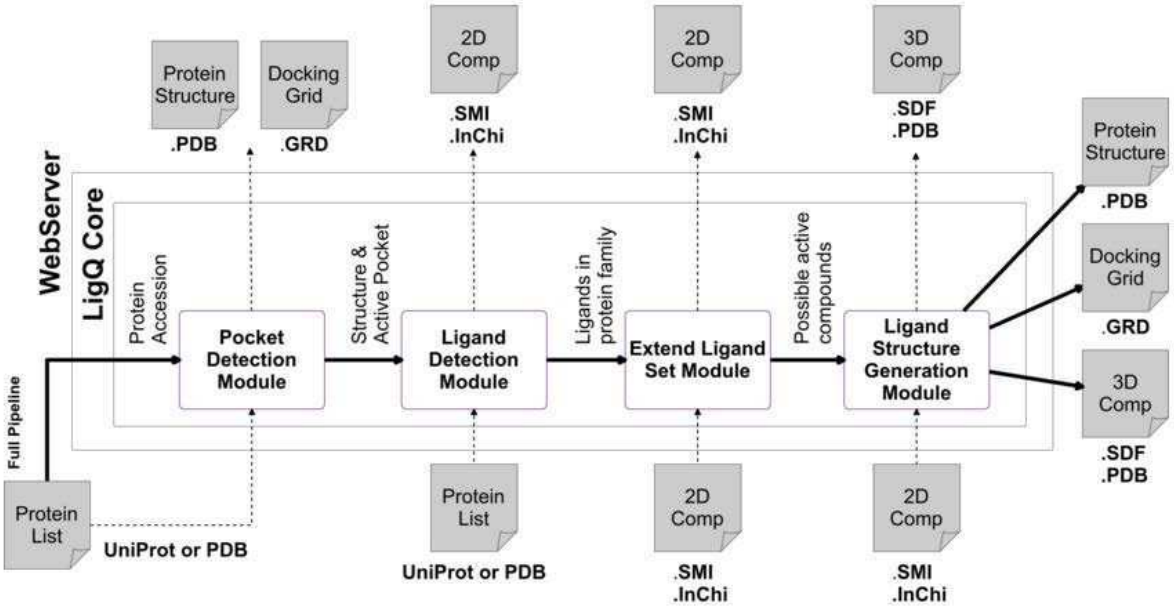


Figure 4.1: Pipeline original de LigQ. El pipeline toma como entrada un identificador UniProt o PDB de una proteína. El resultado final del procesamiento es una lista de estructuras de moléculas pequeñas (ligandos candidatos) y la estructura de la proteína con su correspondiente grilla, delimitando el sitio activo para realizar experimentos de docking. *Fuente: trabajo original de LigQ[37]*

.

Módulo de detección de bolsillos (*Pocket Detection Module*): toma como entrada el código PDB o UniProt de una proteína. En caso de ingresar un identificador de Uniprot, se buscan todos sus cristales asociados en PDB, seleccionándose para posteriores análisis aquel de mayor cobertura de la secuencia total de la proteína. En caso de no existir dicho cristal, el módulo intentará realizar un modelado por homología de la proteína[e]. Luego, una vez definida la estructura tridimensional, se la utiliza como entrada de fpocket, para la

[d] Para dar una idea de la dimensión del software LigQ, un rápido cálculo arroja que su código fuente cuenta con más de 500.000 líneas de código.

[e] Para una descripción del proceso, ver en Apéndice A Subsección A.1.1 "Modelado comparativo (basado en homología)"

detección de sus bolsillos. Los archivos generados por el programa fpocket son guardados en un directorio de salida específico para esta etapa. **Sin embargo, si bien habiendo estado originalmente planificado, estos datos no llegaron a incorporarse al pipeline principal durante el desarrollo LigQ. Nuestro trabajo viene a realizar esta tarea e integrar definitivamente la información de cavidades en la obtención de ligandos candidatos.**

Módulo de detección de ligandos (*Ligand Detection Module*): intenta detectar ligandos de los cuales se tiene alguna evidencia experimental de acoplamiento con la proteína de interés. El espacio de búsqueda de moléculas está definido por: todas las moléculas que han sido co-cristalizadas con la proteína, o con alguna proteína de la misma familia (PFam); todas las moléculas que en ChEMBL que tengan al menos un ensayo marcado como activo con la proteína, o con alguna proteína de la misma familia. LigQ se refiere a este conjunto de compuestos como conjunto semilla (en inglés, *seed*).

Módulo de extensión de ligandos (*Extend Ligand Set Module*): busca enriquecer el conjunto semilla generado en el modulo de detección de ligandos. En el contexto de Virtual Screening, se puede decir que un conjunto de compuestos se encuentra "enriquecido" cuando, al agregarle nuevos ítems, la cantidad de verdaderos ligandos para la proteína objetivo (compuestos que efectivamente se acoplarán a ella) ha aumentado en relación a la cantidad total de compuestos del conjunto original. Al momento de ejecutar el módulo, el usuario define un umbral de similitud, a partir del cual se busca en la base de datos de LigQ compuestos "parecidos"[f]. Este conjunto extendido de compuestos no presenta ya evidencias de acoplamiento con la proteína de interés, y es precisamente eso lo que aporta la posiblidad de encontrar compuestos novedosos.

Módulo de generación de estructuras (*Ligand Structure Generation Module*): para llevar a cabo los posteriores experimentos de docking, virtual screening, o incluso experimentos in-vitro, los investigadores necesitan conocer cuáles son las estructuras tridimensionales que pueden adoptar cada una de los ligandos candidatos. Dado que la información procesada por los anteriores módulos no provee información tridimensional de las moléculas obtenidas, este módulo calcula las estructuras tridimensionales más probables utilizando la biblioteca JChem[64].

4.3.2 Arquitectura modificada

En esta sección detallaremos las decisiones de diseño que tomamos para poder incorporar nuestro sistema de recomendación al esquema anterior.
Como primer punto es importante recalcar que, con la idea de poder explotar al máximo las funcionalidades de la herramienta, decidimos no modificar la arquitectura original, sino más bien adaptar nuestros esquemas a ésta. Es por ello que la predicción de nuestro modelo se integra al pipeline de la misma forma que los distintos módulos se comunican entre ellos: mediante archivos. En nuestro caso, definimos un nuevo tipo de salida: un archivo con extensión `.recsys`.

[f] La medida de similitud se calcula mediante el índice de Tanimoto[93].

Como fue explicado anteriormente, la etapa de detección de ligandos (denominada internamente `computeSeed`) tiene como objetivo computar un conjunto de compuestos que se relacionan directamente con la proteína en cuestión. Cuenta con 2 subetapas: `searchInCrystals`, en donde se buscan compuestos existentes en cristales y `searchInBioAssays`, en la cual se buscan compuestos existentes en ensayos. Para cada una de ellas se generan archivos de salida con el listado de compuestos elegidos en cada caso, los cuales son utilizados posteriormente por el módulo de extensión de ligandos (internamente, `extendSeed`). Nuestro objetivo fue entonces añadir la etapa `searchInRecommendationSystem`, en donde se buscan compuestos en nuestro sistema de recomendación, y generar el archivo de salida correspondiente. Este archivo contiene la salida de la predicción para cada pocket de la proteína, en forma de una lista de IDs de compuestos, precedido por "RECSYS" (ejemplo `'RECSYS O60885 HG8'`), como puede verse en la Figure 4.2.

```
computeSeed.bioassays (extracto)

experiment  targetID  compID
------------------------------
BIOASSAY    O60885    CHEMBL646
```

```
computeSeed.crystals (extracto)

experiment  pdbID     compID
------------------------------
CRYSTAL     5Z5U      96U
```

```
computeSeed.recsys (extracto)

experiment  targetID  compID
------------------------------
RECSYS      O60885    HG8
```

Figure 4.2: Ejemplos de archivos de salida del módulo de detección de ligandos (`computeSeed`) para la proteína UniprotID 060885. Uno de los cristales encontrados tiene pdbID 5Z5U. La tercera columna contiene el potencial ligando, indicado por su ID de ChEMBL en `.bioassays` y de PDB, en los dos restantes.

.

Propiedades geométricas de las cavidades (MVEE)

Para poder aplicar el método `predict()` del estimador se necesita contar con un vector de características de los bolsillos de las proteínas. Dado que dichas características no siempre se encuentran precalculadas en la base de datos que posee LigQ, las mismas deben ser computadas en tiempo de ejecución.

En el caso de las características volumétricas de los bolsillos, la información la tuvimos que recopilar de la etapa de detección de bolsillos (`findPockets`). En la figura 4.3 se puede observar un ejemplo del archivo de configuración de la etapa de detección de ligandos (`computeSeed.cfg`). En el mismo se detalla el directorio en donde se encuentran los resultados de la corrida de fpocket (`InPath`). Utilizamos la información del archivo `pqr`, alojado en dicha carpeta, para generar un MVEE (ver sección 2.2.6.1) con el fin de obtener los datos correspondientes a los radios que definen a cada bolsillo.

```
[general]
InPath=/tmp/output/fpocket/O60885_out
targetID=O60885
command=computeSeed
runID=computeSeed
outputFolder=/tmp/output/
email=example@fcen.uba.ar
targetsOrigin=UNIPROT
```

Figure 4.3: Ejemplo de archivo `computeSeed.cfg`, generado para la etapa de detección de ligandos de LigQ.

.

Las funciones utilizadas para la generación de MVEE se encuentran en el archivo

`computeSeed_Functions.py`, que a su vez utiliza métodos declarados en `pocket/Fpocket.py` para manejo de archivos `pqr` y en el módulo `recsys/`. En la figura 4.4 se puede observar un detalle de la disposición de los nuevos módulos en la jerarquía de directorios de LigQ.

```
- (project root directory)
 - ... (docker related files)
 - thid-party/
 - required-files/
 - sbg/
  - ... (project files)
  - sbgserver/
  - tests/
        - sbg/
         - ligQ_Pipeline/  # aquí figuran todos los módulos de ligQ
         - recsys/
          - _init__.py
                - engine.py  # clases y métodos utilizados en la predicción
                - utils.py  # funciones de procesamiento de archivos .csv/.joblib
                - data/
                 - predictor.joblib  # modelo predictor
                 - clusters.csv  # clusters de ligandos existentes
                 - balltree1.joblib  # modelos para el calculo de distancias
                 - balltree2.joblib
```

Figure 4.4: Árbol de directorios del nuevo proyecto LigQ 2.0.

En particular, a fin de permitir en el futuro la incorporación de nuevas features de forma sencilla, se definió un mecanismo que utiliza el patrón de diseño Decorador[94]. El mismo otorga la posibilidad de poder extender la funcionalidad de un método sin tener que alterar su código fuente, actuando como *wrapper* de su entrada y salida, como puede verse en la Figure 4.5.

```
@_add_features(FPocket.get_mvee_features_from_pqr_file)
@_add_features(FPocket.get_fpocket_attributes_from_pqr_file)
def get_pocket_attributes(filepath):
    '''
    Takes protein's fpocket *.pqr output for pocket, i.e.:
        /.../fpocket/O60885_out/pockets/pocket0_vert.pqr

    Then returns:
        (feature_1, feature_2, etc)
    '''

    return {}
```

Figure 4.5: Función para obtener las features de pockets que luego serán tomadas como entrada por el modelo para generar la predicción.

Finalmente, un esquema de la nueva arquitectura puede verse en la Figure 4.6.

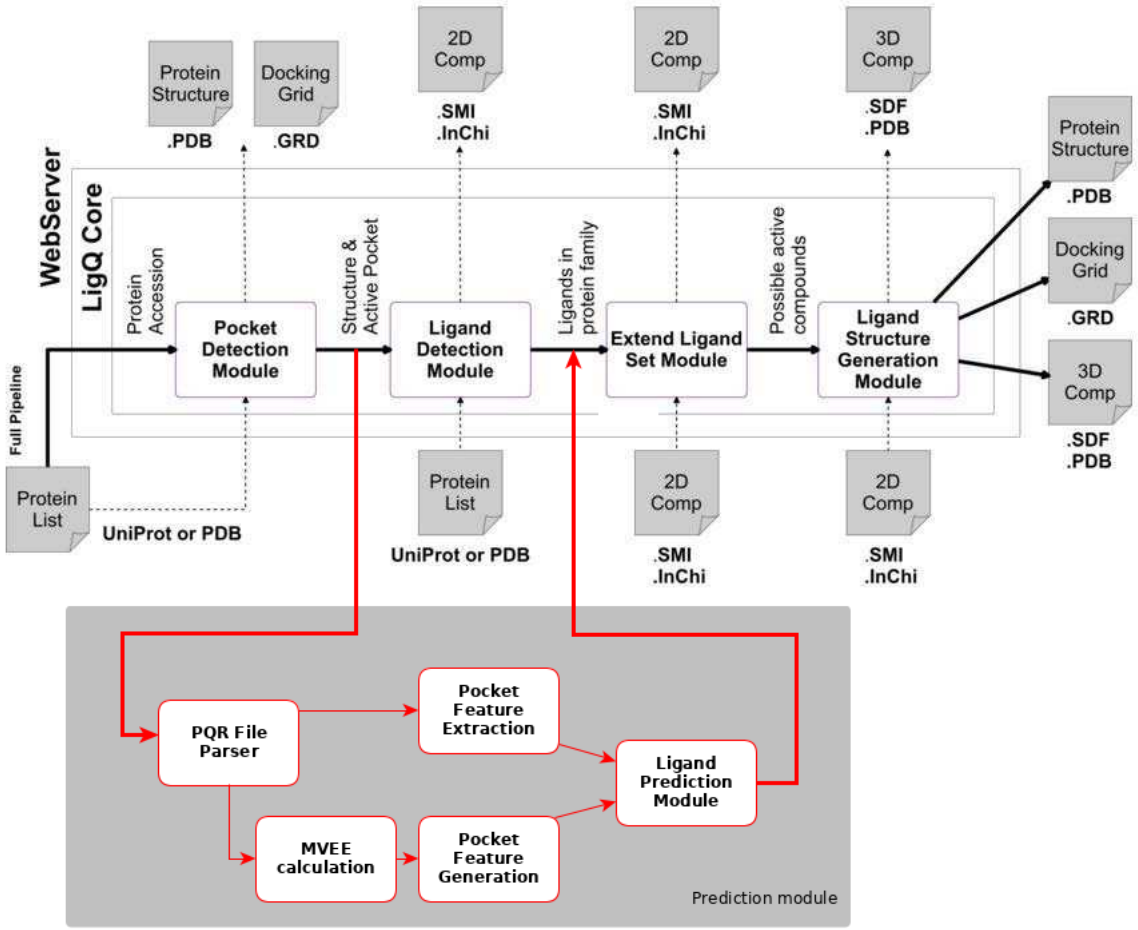


Figure 4.6: Pipeline modificado de LigQ

4.3.2.1 Predictor elegido y modo de uso

Tanto el predictor elegido (la combinación de hiperparámetros final se encuentra detallada en la section 3.6) como también los modelos de escalado utilizados y los BallTree de cada cluster fueron persistidos en archivos `joblib`, siguiendo las recomendaciones de scikit-learn[g].
Los archivos `csv` del clustering de ligandos permiten obtener el cluster al cual corresponde el ligando.
Los modelos de BallTree para cada cluster permiten calcular la distancia a un ligando ideal de manera rápida.

[g]En el apartado de Model persistence, en el URL: `https://scikit-learn.org/stable/modules/model_persistence.html`, se deja claro lo siguiente: *"In the specific case of scikit-learn, it may be better to use joblib's replacement of pickle (dump & load), which is more efficient on objects that carry large numpy arrays internally as is often the case for fitted scikit-learn estimators[...]"*

El pipeline cuenta con un módulo con métodos específicamente diseñados para cargar los modelos desde los archivos correspondientes y generar una predicción *on-the-fly*.

4.4 Conclusiones

En esta etapa hemos ampliado las capacidades de Virtual Screening de la herramienta LigQ con modelos de Machine Learning que predicen ligandos con buenas perspectivas de acoplamiento con pockets de una proteína objetivo. Este desarrollo supone, a la vez, un cierre del circuito originalmente ideado para LigQ y un piso a partir de cual seguir extendiendo la herramienta.
Por último, una de las famosas citas de Robert Baden-Powell (actor, pintor, músico, militar, escultor, escritor y fundador del Movimiento Scout) dice: "Leave this world a little better than you found it." ("Deja a este mundo en mejores condiciones en las que lo encontraste."). Abrazando esta hermosa idea encaramos el desarrollo de las nuevas funcionalidades sobre LigQ. Una breve lista de los *extras* surgidos en este devenir fueron: *bug fixing*, *refactoring*, *linting*[h], *performance* y *testing*, hacer de LigQ un sistema *cross-plataform* gracias a la migración a *docker containers*. Montamos el código en un sistema de versionado[i] y lo dejamos disponible para acceso público[j].

[h] Es el proceso de verificación de potenciales errores lógicos en el código fuente, así como también discrepancias en el formateo (por ejemplo, indentar mezclando tabulaciones y espacios) y la no adherencia a convenciones de codificación (por ejemplo, PEP 8 de Python [95])

[i] Se encuentra alojado en un `https://git.exactas.uba.ar/pturjanski/LigQ`, un repositorio privado.

[j] Actualmente se encuentra en línea en `http://ligq2.qb.fcen.uba.ar/`.

5 | Predicción ligando-proteína

Desde un punto de vista general, nuestro dataset se encuentra compuesto por features de pockets, features de ligandos, y sus interacciones. En base a este dataset, desarrollamos un modelo de aprendiza automático para predecir la relación proteína⇒ligandos. Si bien, desde un punto de vista estrictamente computacional, la relación de acoplamiento puede considerarse bidireccional, surge de manera natural transitar el camino inverso: desarrollar un modelo de aprendizaje automático que prediga la relación ligando⇒proteínas. Con este modelo intentaremos responder la pregunta *"Dado un compuesto, ¿a qué proteínas puede acoplarse?"*
La pregunta no es para nada trivial. Dos de las mayores razones por la cual muchos compuestos no llegan al final del proceso de desarrollo de drogas (ver en capítulo 1, figura 1.2), son su toxicidad y que los mismos exhiben baja eficacia clínica. Esto se debe a que las proteínas actúan en complejas redes interconectadas. En el primer caso, un compuesto puede tener efectos adversos si, por ejemplo, interactúa con proteínas distintas a la objetivo, impactando de manera negativa en el organismo. En el segundo caso, contrario al paradigma dominante de diseñar ligandos con máxima selectividad que actúen en blancos proteicos específicos, se ha descubierto que muchas drogas eficaces actúan por modulación de múltiples proteínas[96].
Entre sus potenciales usos, poder identificar de manera confiable a qué proteína puede acoplarse un ligando tiene múltiples aplicaciones[31]:

- Identificar proteínas objetivo para un compuesto que causa un efecto en el organismo, y cuyo de mecanismo de acción es desconocido.
- Identificar proteínas nuevas que se acoplan a un compuesto.
- Utilizar una droga para otra enfermedad distinta para la cual fue desarrollada y aprobada.

Para dar respuesta a estos problemas, desarrollamos un sistema de recomendación con una arquitectura similar al desarrollado en el Capítulo 3. Para ello decidimos incorporar la información proveniente de ChEMBL dado que los gráficos de interacciones proyectadas originales (basados únicamente en PDB) resultaban poco convenientes para esta etapa, dado que la cantidad de ligandos era muy baja. Al sumar la información de ChEMBL y realizar nuevamente el análisis, los resultados mostraron una leve mejoría, lo cual nos orienta a tomar este nuevo dataset como referencia para esta nueva etapa. También consideramos utilizar el dataset 8L (descripto en la sección 2.2.8.1), pero poseía una cantidad mucho menor de ligandos, lo cual podía afectar los resultados. En última instancia, por cuestiones de extensión de este trabajo de tesis no realizamos comparaciones de resultados entre ambas variantes, pero queda como trabajo futuro, al igual que el enriquecimiento de los datos de interacciones ligando-proteína con nuevas fuentes de datos.
Dado que el problema presenta la misma estructura que el problema de recomendación proteína → ligandos, decidimos utilizar las mismas técnicas para cada una de las etapas. En este sentido, podemos argumentar que el problema se traduce principalmente en invertir cuáles son las propiedades que funcionan como datos de entrada y cuáles son las que se desea inferir, y ajustar los parámetros de los modelos al nuevo comportamiento. El pipeline propuesto se detalla en la figura 5.1.

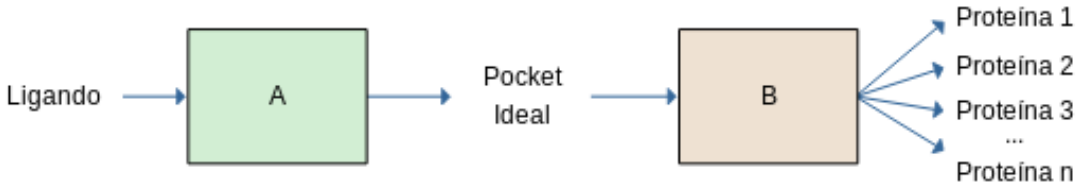


Figure 5.1: En la etapa A, para un ligando de entrada se calcula el pocket *ideal* que mejor chance tienen de acoplarse al mismo. En la etapa B se buscan mediante clustering aquellos pockets del dataset completo (archivo `unified_pocket_id.csv`) más similares al pocket ideal. Luego de ello, se buscan las proteínas de las cuales estos pockets son parte.

Para el modelo de clustering sobre los datos de `unified_pockets.csv`, fuimos variando el parámetro k de K-means y graficamos la inercia del clustering total. En la figura 5.2 se ve que, a partir de un valor de $k = 150 - 200$, la curva queda acotada inferiormente por un valor de inercia en torno a 100.000.

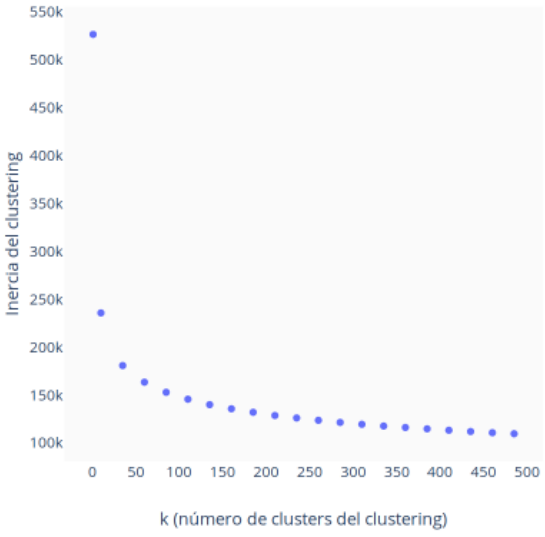


Figure 5.2: Inercia del clustering de `unified_pockets.csv` a medida que aumentamos el parámetro k de K-means.

5.1 Obtención del pocket *ideal*

5.1.1 Redes Neuronales aplicadas a regresión

Ajuste del modelo usando Grid Search

A continuación listamos los rangos de valores de hiperparámetros, a partir de los cuales se construirán todas las combinaciones en la etapa de Grid Search, con los que se entrenarán las distintas redes neuronales.

Hiperparámetros	**Valores**
activation	tanh
batch_size	40
dropout_rate	0.0, 0.2
epochs	50, 100
hidden_layer_extension	(4, 10), (8, 8), (15, 15)
init_mode	lecun_uniform
input_size	10
learn_rate	0.001, 0.01
neurons	13
optimizer	ADAM
output_size	16
weight_constraint	2

Table 5.1: Valores de hiperparámetros utilizados durante la etapa de Grid Search.

Evaluación de rendimiento: métricas y gráficos

Los valores de error MSE (la misma métrica utilizada en el Capítulo 3 para evaluar el rendimiento de los modelos de regresión proteína⇒ligando) correspondientes a las configuraciones listadas en la tabla 5.1 fueron todos muy similares. Los resultados pueden observarse en la tabla 5.2.

Epochs	Dropout	HiddenLayers	LearnRate	Puntaje
50	0.0	(4, 10)	0.001	-0.31590571
50	0.0	(4, 10)	0.0001	-0.31909322
50	0.0	(8,8)	0.001	-0.31022554
50	0.0	(8,8)	0.0001	-0.31443856
50	0.0	(15,15)	0.001	-0.34096871
50	0.0	(15,15)	0.0001	-0.30868432
100	0.0	(4, 10)	0.001	-0.31170992
100	0.0	(4, 10)	0.0001	-0.31330481
100	0.0	(8,8)	0.001	-0.30658757
100	0.0	(8,8)	0.0001	-0.30840816
100	0.0	(15,15)	0.001	-0.34087023
100	0.0	(15,15)	0.0001	-0.3013903
50	0.2	(8,8)	0.001	-0.32297382
50	0.2	(8,8)	0.0001	-0.3279089
50	0.2	(15,15)	0.001	-0.34529321
50	0.2	(15,15)	0.0001	-0.32234739
100	0.2	(8,8)	0.001	-0.32227209
100	0.2	(8,8)	0.0001	-0.32113427
100	0.2	(15,15)	0.001	-0.34107499
100	0.2	(15,15)	0.0001	-0.32060711

Table 5.2: Valores promedio de MSE obtenidos de la corrida de Grid Search. Por cuestiones de capacidad de cómputo, se realizaron corridas separadas para ciertas combinaciones, quedando algunas pocas sin probar.

Los resultados de la tabla 5.2 muestran, en todos los casos, valores muy similares. El hecho de que distintos modelos logren una performance similar puede estar relacionados a varios factores, entre ellos que el conjunto de features escogido introduce ruido a la hora entrenar el modelo. Es por ello que a continuación recurrimos a la técnica de *feature importance* para entrenar a las redes utilizando sólo un subconjunto de features de entrada (las de mayor relevancia en función de la predicción).

Feature Importance

Al igual que el análisis de *feature importance* realizado para la predicción proteína-ligandos, aplicamos esta técnica al dataset total (*sesgado*). Los resultados obtenidos pueden verse en la tabla 5.3, en donde claramente se nota que las tres primeras propiedades son las de mejor puntaje. El resto de las features tiene puntaje igual a cero.

Feature de ligando	Importancia (Total)
logp	0.8232951429361403
arom_rings	0.1756128986615215
nrb	0.0010919584023382448
mwt	0.0
hbd	0.0
hba	0.0
tpsa	0.0
lr	0.0
mr	0.0
sr	0.0

Table 5.3: Features para caracterizar a los ligandos junto a sus respectivos valores de importancia obtenidos a ravés de la técnica de feature importance.. Dichos valores corresponden al dataset (*sesgado*).

En función de los resultados de feature importance hicimos corridas de Grid Search, entrenando a las redes neuronales con configuraciones de hiperparámetros similares a las ya utilizadas anteriormente, pero utilizando como dataset de entrada el dataset sesgado restringido sólo a los 3 features más relevantes (ver 5.3). Las configuraciones utilizadas durante la etapa de GridSearch pueden verse en la tabla 5.4.

Hiperparámetros	**Valores**
activation	tanh
batch_size	40
dropout_rate	0.0
epochs	40
hidden_layer_extension	(variable)
init_mode	lecun_uniform
input_size	3
learn_rate	0.01, 0.0001
neurons	13
optimizer	adam
output_size	10
weight_constraint	2

Table 5.4: Valores de hiperparámetros utilizados durante la etapa de Grid Search.

De la corrida de Grid Search, en base a las configuraciones de la tabla 5.4, la que obtuvo mejor resultado utiliza `hidden_layer_extension` igual a `(4, 1)`, y `learn_rate` igual a 0.0001. Su puntaje fue de -0.2117655390766483, lo cual afortunadamente mejora la performance de los modelos anteriores que utilizan la totalidad de las features. Al igual que en el caso de los modelos de predicción proteína-ligandos, recurrimos a la curva de aprendizaje, que mide la performance del modelo en función de una cantidad creciente de muestras.

Graficamos en rojo los puntajes correspondientes al MSE promedio para los casos de entrenamiento y análogamente en verde para los casos de validación. La gráfica para este caso puede verse en la figura 5.3. En ella puede verse una curva interesante, en donde tanto los casos de entrenamiento como los de validación van mejorando su rendimiento con una mínima diferencia entre ambas. Esta pequeña diferencia usualmente es denominada *generalization gap*.

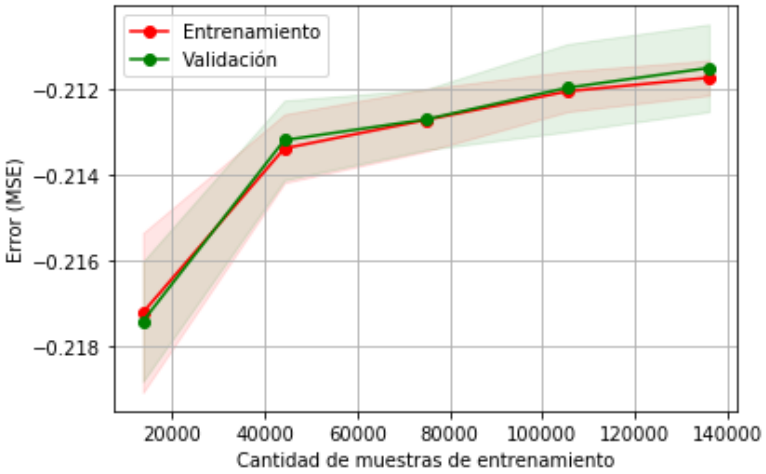


Figure 5.3: Curvas de aprendizaje para la mejor modelo de red neuronal obtenido a través de la técnica de GridSearch (valores de búsqueda en la tabla 5.4), en base al Dataset sesgado sólamente con los 3 features más relevantes.

Debido a que este modelo mostró una mejor performance que los anteriores, decidimos adoptarlo como modelo definitivo para esta etapa.

5.2 Obtención de los pockets reales

Una vez finalizado el entrenamiento del modelo de regresión, de forma análoga a la recomendación proteína-ligandos, con el vector de features de un ligando como entrada del modelo, predecimos el vector de features del pocket *ideal*. En la siguiente etapa, procedemos a buscar entre el total de pockets de nuestro dataset (archivo `unified_pockets.csv`) los n pocktets más similares al pocket ideal, para un valor de n preestablecido (ver Figure 5.1, etapa B de nuestro pipeline). Para ello, primero clasificamos los pockets mediante técnicas de clustering. Luego, utilizando el mismo modelo de clasificación, predecimos la clase del pocket ideal. Al igual que en el caso anteriormente mencionado, decidimos utilizar la técnica K-means para realizar el clustering de los pockets, cuyas features fueron previamente escaladas como se indica en la sección 3.2.1.
Por otro lado, una vez conseguidos los pockets "similares" al pocket de entrada, mediante el archivo `annotated_pockets.csv` obtenemos el conjunto de pdbIDs de las estructuras proteicas que se corresponden a esos pockets. El output final del pipeline es, entonces, un conjunto de IDs de estructuras de la base de datos RCSB.

5.3 Evaluación de rendimiento global

Evaluación de rendimiento: exactitud de pertenencia a un cluster (métrica 1)

Al igual que en el esquema de recomendación del capítulo 3, nos interesa conocer qué proporción de los pares ligando-pocket de la primera muestra de control tienen coincidencia entre el cluster al cual pertenecía cada pocket real de un ligando y el cluster determinado para su pocket ideal. A diferencia del caso anterior, esta vez escogimos una única combinación de dataset y escalado,

debido a que desde el comienzo optamos por tomar únicamente el dataset *sesgado*. Por otro lado, no fue requerido un doble escalado debido a que todos los pockets presentes en `pockets.csv` poseen interacciones con los ligandos. Primero predecimos el pocket ideal de un ligando del conjunto, y luego, utilizando el modelo de clustering que generamos previamente, determinamos a qué cluster (identificado por un `cluster_id`) pertenecen, por un lado, los pockets reales y, por otro, el pocket ideal. Finalmente, calculamos la cantidad de coincidencias entre el `cluster_id` del pocket ideal y el `cluster_id` de los correspondientes pockets reales. Repetimos el procedimiento con todos los ligandos, y, de manera análoga a lo realizado en la evaluación del recomendador pocket-ligandos, calculamos el valor de *accuracy* (métrica 1). El resultado puede verse en la tabla 5.5.

Características del Dataset	Métrica 1 (Porcentaje)
Dataset sesgado, un escalado	0.043680 (4, 37%)

Table 5.5: Valores de accuracy, en la predicción de cluster de pertenencia.

Evaluación de rendimiento: coincidencia en al menos un cluster (métrica 2)

A continuación medimos la proporción de coincidencias entre el cluster del pocket ideal con al menos un cluster de los pockets reales. Con la información sobre clusters generada para el cálculo de la métrica 1, evaluamos para qué proporción de los ligandos existía dicha coincidencia. Esta información puede verse en la tabla 5.6.

Características del Dataset	Métrica 2
Dataset sesgado, un escalado	6.49%

Table 5.6: Porcentaje de coincidencias entre el cluster del pocket ideal de cada ligando con al menos uno de los clusters de los pockes reales del ligando.

En este caso, la proporción de aciertos a nivel global (definidos por la pertenencia al mismo cluster) es baja. A continuación presentamos la evaluación del rendimiento del sistema pero a nivel intracluster.

Evaluación de rendimiento utilizando la curva ROC y ROC AUC

De la misma manera que fue realizado para la evaluación del modelo proteína-ligandos, en este caso, para cada uno de los ligandos del conjunto de test, nuevamente, acotados a aquellos con los que se obtuvieron predicciones correctas según la métrica 2 ("subconjunto ROC M2"), graficamos la curva ROC y calculamos el ROC AUC. En la tabla 5.7 exponemos su valor promedio para el pipeline elegido.

Características del Dataset	ROC AUC promedio (desvío estandar)
Dataset sesgado, un escalado	0.52 ($\pm$ 0.17)

Table 5.7: ROC AUC obtenido para las recomendaciones de pockets de ligandos pertenecientes a la primera muestra de control.

Nos encontramos un valor promedio de ROC AUC cercano al 0.5, lo que sugiere que el recomendador no funciona mejor que seleccionar los valores al azar. Sin embargo, el desvío estandar puede indicar que la existencia de casos para los que sí haya un mejor desempeño. Analizamos con qué proporción de ligandos del subconjunto "subconjunto ROC M2" se obtenía ROC AUC mayores que 0.6, 0.7, 0.8 y 0.9. Los resultados pueden verse en la tabla 5.8. Lamentablemente en ella los resultados no fueron muy buenos, con cerca del 21% de los casos consigue ROC AUC $>$ 0,6 con respecto al subconjunto de ligandos sobre el que realizamos las mediciones.

Características del Dataset	>0.6	>0.7	> 0.8	> 0.9
Dataset sesgado, un escalado	20.92%	5.66%	4.48%	4.27%

Table 5.8: Proporción de los casos estudiados para los cuales ROC AUC es mayor a los valores indicados.

Sin embargo, el rendimiento del modelo debemos evaluarlo con respecto a la totalidad de los ligandos de la muestra. Por lo que, considerando el porcentaje indicado en la tabla 5.6, concluimos que el modelo produjo muy buenas predicciones (ROC AUC $>$ 0.9) en un 0.28% del total de los ligandos de la muestra de control. Esto se interpreta de la siguiente forma: dada las features de un ligando, se ingresan sus features como entrada del modelo; entre los pockets recomendados, sólo contamos con 0.28% de probabilidad de que exista uno al cual se pueda acoplar el ligando de entrada.

Discusión

Resulta interesante sin embargo mencionar que, gracias a los avances en técnicas de High Throughput Screening[97], que permite la automatización de experimentos a gran escala, pueden realizarse experimentos de interacciones con target biológicos (como proteínas) en el orden de millones de reacciones en cuestión de horas[98]. Teniendo esto en consideración, si bien los resultados de rendimiento de los modelos distan de ser los ideales, cabe preguntarse si tengan un grado de aporte más bien moderado dado el contexto del problema del área.

5.4 Modelos elegidos: configuración

Finalmente, las configuraciones de parámetros elegidas de redes neuronales y K-means se definen a continuación:

- Se toman únicamente las features `logp`, `arom_rings` y `nrb` para los ligandos.
- Para la red neuronal

```
{activation: 'tanh',
 batch_size: 40,
 dropout_rate: 0.0,
 epochs: 40,
 hidden_layer_extension: (4, 1),
 init_mode: 'lecun_uniform',
 input_size: 3,
 learn_rate: 0.0001,
```

```
neurons: 13,
optimizer: 'adam',
output_size: 16,
weight_constraint: 2}
```

- Para K-means, optamos por un clustering de pockets con $k = 200$ y los restantes parámetros por defecto.

5.5 Conclusiones

Tomando como referencia el desarrollo hecho en la etapa de recomendación proteína-ligandos, nuevamente nos focalizamos en las redes neuronales como técnica de regresión, dado que cuentan con una gran cantidad de opciones a la hora de configurar los modelos. Por otro lado, ya conociendo las particularidades de los datos y sus distribuciones, optamos replicar los esquemas de validación previamente utilizados. Esto nos permite mantener una base común de evaluación de rendimiento de los modelos

De la misma manera, los scripts de datos utilizados para el análisis, el proceso de escalado de los datos, la generación de las particiones de train/test y el entrenamiento de los modelos utilizados quedan disponibles para eventuales usos posteriores en forma de Jupyter notebooks.

A su vez, estas decisiones permitieron acotar el desarrollo de este trabajo de tesis. Queda como un interesante trabajo a futuro modificar estos esquemas e inclusive los datasets utilizados para atender a las particularidades del problema de *Inverse Virtual Screening*.

Finalmente, podemos decir que este trabajo introduce una nueva línea de investigación en la Plataforma Bioinformática Argentina, lo cual deja muchas expectativas de cara al futuro.

6 | Implementación de la herramienta Reverse-LigQ

Como segundo objetivo del presente trabajo, se propuso desarrollar una herramienta que funcione en el sentido inverso al que lo hace actualmente LigQ. Es decir, implementar una herramienta que tome como entrada una molécula y arroje como salida a aquellas proteínas que tengan buenas chances de acoplamiento con dicha molécula. Si bien analizamos la posibilidad de incorporar este nuevo circuito a LigQ, se llegó rápidamente a la conclusión de que resultaba poco práctico por varios motivos. En primer lugar, porque el pipeline de LigQ se ejecuta de manera secuencial, utilizando la salida de cada paso como entrada del paso siguiente. En este sentido, este nuevo circuito plantea en sí mismo un pipeline completamente distinto. Sumado a esto, el fuerte acoplamiento de los componentes de LigQ (problema detallado en section 4.2) dificultaba introducir nuevas funcionalidades en el sistema. Finalmente, nos inclinamos por definir un tipo de diseño frontend-backend incompatible con la arquitectura actual de LigQ. La idea de este esquema es que permita la ejecución de consultas remotas a través de una API REST, sin requerir acceso al código fuente, lo cual resulta imposible de alcanzar con la implementación actual de LigQ. Por último, dado que no todas las dependencias de LigQ pudieron actualizarse a su última versión, estaríamos forzándonos a trabajar con tecnología desactualizada, lo cual se busca evitar dentro de lo posible para cualquier desarrollo.
Todas estas razones nos llevaron a desarrollar una herramienta independiente, a la que denominamos `Reverse LigQ`. En las secciones subsiguientes detallaremos todo lo relacionado a esta nueva herramienta.

6.1 Decisiones de diseño

Para el nuevo desarrollo tomamos las siguientes decisiones en cuanto a cuestiones de arquitectura de software, diseño de los componentes, usabilidad, modularidad y extensibilidad:

- *Containerización* de la aplicación: al igual que en LigQ, para que la instalación y el uso de la herramienta sean independientes del entorno de ejecución (multiplataforma), facilitando así su distribución y haciendo que resulte práctica la puesta en marcha a los administradores de sistemas.

- Separación en capas de presentación (*frontend*) y procesamiento (*backend*): este esquema permite obtener los resultados del procesamiento de forma programática, sin necesidad de tener que hacerlo a través de una página web. Esto puede resultar muy útil para que desarrollos posteriores que quieran utilizar los modelos de predicción de ligando a proteína puedan crear scripts que se comuniquen fácilmente con la herramienta. A su vez, si se quiere reemplazar el *backend* por una nueva implementación más performante o con un predictor más sofisticado, puede hacerse perfectamente sin por ello tener que modificar la capa de presentación (frontend), a la que los futuros usuarios están acostumbrados.

- Comunicación entre ambas capas a través de API REST: fundamentalmente para utilizar un protocolo estándar, fácil de adoptar por cualquier cliente.

- Datos ingresados por el usuario: las propiedades del compuesto (con excepción de los descriptores geométricos) son introducidas por el usuario de forma manual. Esto es una ventaja dado que no todas las bases de datos cuentan con la información completa de un compuesto, de forma que el usuario puede consultar diversas fuentes para obtener los valores que más considere apropiados. El archivo SDF con la estructura del compuesto, requerido para calcular el MVEE, es obtenido de forma transparente al usuario en el *backend* (se explica en detalle en "Descriptores geométricos del compuesto" de esta misma sección).
  El sistema realiza, además, una validación de tipos de datos, notificando al usuario si los valores no se corresponden con los necesarios para ese descriptor en particular.

- Tecnologías: el *frontend* fue desarrollado en Angular[a] v8.3.8 y el *backend* en lenguaje Python 3, utilizando las bibliotecas Falcon[b] v2.0.0 para la API y Cerberus[c] v1.3.1 para la validación de datos de entrada. Por último, aclaramos que la implementación del *backend* cuenta con una restricción de la cantidad de solicitudes por segundo (*rate limit*) para evitar consultas masivas que puedan sobrecargar al servidor y, el *frontend* por su parte, incorpora un validador reCaptcha[d], por el mismo motivo.

Descriptores geométricos del compuesto

Al igual que lo que sucede con las cavidades en la nueva implementación de LigQ , las features que definen la forma del compuesto objetivo se calculan a partir de su información estructural. Para lograr ello, el sistema utiliza el `ID` del compuesto provisto por el usuario para realizar una consulta a RCSB[e], con el fin de obtener un archivo SDF, el cual describe su geometría tridimensional. Como se detalló en el chapter 2, correspondiente a la generación del conjunto de datos, el sitio RCSB ofrece dos alternativas de descarga: una que describe la estructura "ideal" y otra con la estructura extraída de un modelo de PDB (*model*), que es por la que finalmente optamos. La diferencia entre ambas es explicada en el Apéndice A, Sección A.2.11.3.
A partir de estos archivos, el sistema sigue los mismos pasos que en LigQ, utilizando la biblioteca `mvee` para calcular el elipsoide y sus radios. Los métodos para procesar los archivos SDF y generar la predicción se encuentran disponibles en el módulo `backend/pipeline/compound.py`.

Predictor seleccionado y modo de uso

Al igual que en el caso de predicción proteína-ligandos de LigQ, en este caso los modelos de predicción también fueron persistidos en archivos de tipo `joblib`, junto con sus modelos de escalado y BallTree de cada cluster. Las referencias a los mismos se encuentran disponibles en el módulo `backend/pipeline/recommender.py`, en donde además se define la clase `Engine` que implementa los métodos que, haciendo uso de los modelos, generan la recomendación. Los

[a] `https://github.com/angular`
[b] `https://github.com/falconry/falcon`
[c] `https://github.com/pyeve/cerberus`
[d] `https://www.google.com/recaptcha/intro/v3.html`
[e] https://files.rcsb.org/ligands/view/{ID}_{model|ideal}.sdf

métodos principales se llama `recommend_pockets` y `recommend_proteins` y su implementación se puede ver en la Figure 6.1.

```
class Engine():

    ...


    def recommend_pockets(self, ligand_features):

        # Calculate ideal pockets features for this set of ligand features
        ideal_pocket_features = self.predict_pocket_features(
                ligand_features)

        # Calculate ideal pocket cluster
        cluster_label = self.get_cluster_label_of_predicted_pocket(
                ideal_pocket_features)
        pockets_in_cluster = self.get_related_pockets_from_clustering(
                cluster_label)

        # Get similar pockets (K-means + BallTree)
        similar_pockets = self.get_k_closest_neighbors(
            neighborhood=list(pockets_in_cluster),
            target=ideal_pocket_features,
            k=NUMBER_OF_RECOMMENDATIONS,
            cluster=cluster_label)

        # Return a subset of similar pockets
        return similar_pockets

    def recommend_proteins(self, ligand_features):

        recommended_pockets = self.recommend_pockets(ligand_features)

        recommended_proteins = []

        for p in self.pockets_to_proteins:
            if p['unified_pocket_id'] in recommended_pockets:
                recommended_proteins.append(p['pdb_id'])

        return list(set(recommended_proteins))
```

Figure 6.1: Funciones que toman la información del ligando para computar la predicción de pockets candidatos y, posteriormente, la búsqueda de ids de PDB asociados a éstos.

6.2 Arquitectura de la herramienta

El circuito completo de recomendación comienza con el ingreso de los datos correspondientes al ligando por parte del usuario, y finaliza con la devolución de una lista de IDs PDB de las proteínas propuestas. En resumen, las etapas que comprende son las siguientes: validación de

datos ingresados, descarga del archivo SDF, cálculo de MVEE a partir de dicha información, predicción de pocket ideal a partir de los descriptores calculados y los ingresados manualmente, búsqueda de cluster al cual pertenece dicho pocket, selección de pockets reales similares por cercanía y, finalmente, correspondencia de los pockets a proteínas, de las cuales se devuelve su PDB ID. Un esquema del pipeline puede visualizarse en la Figure 6.2.

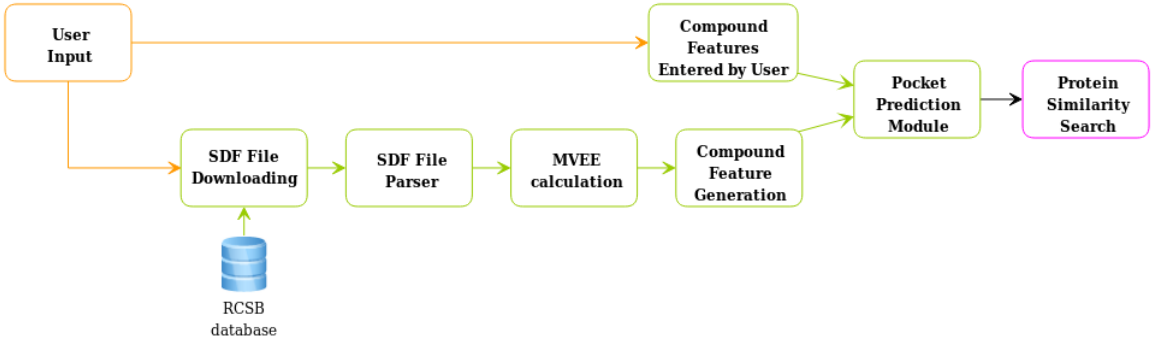


Figure 6.2: Esquema del funcionamiento del pipeline de Reverse-LigQ

6.3 Conclusiones

Junto con el trabajo hecho sobre LigQ, el desarrollo de esta herramienta termina de constituir una base sólida para el análisis de interacciones proteína-ligandos, a partir de la cual futuras investigaciones puedan nutrirse. Al igual que lo hicimos con LigQ, el espíritu de este desarrollo busca no sólo garantizar accesibilidad, sino también su extensibilidad y futuras mejoras. Es por ello que también el código se encuentra disponible bajo el sistema de versionado[f]. El sistema, a su vez, queda disponible para acceso público[g].

[f] Está alojado en un `https://git.exactas.uba.ar/pturjanski/reverse-ligq`, un repositorio privado.
[g] Actualmente se encuentra en línea en `http://reverse-ligq.qb.fcen.uba.ar/`.

7 | Conclusiones generales

En el presente trabajo recolectamos información de bases de datos públicas de proteínas y compuestos químicos para construir un dataset de calidad que sirva a los fines de *Virtual Screening* e *Inverse Virtual Screening*. Para llevar a cabo esta tarea, el dataset inicial fue sometido a una etapa de curado y extensión, lo que dio lugar a una amplia gama de análisis estadísticos y modificaciones. En este sentido, se describió detalladamente los pasos seguidos para su confección y análisis, de manera que dicha información pueda resultar de mucha utilidad para trabajos futuros.
Por otro lado, el dataset final contribuye al área de la problemática con información inexistente en la actualidad en estas bases de datos públicas: descriptores volumétricos que definen la forma de los bolsillos de una proteína y de los compuestos que, presumiblemente, pueden acoplarse a ella. Este desarrollo supone un aporte novedoso al estudio general de la bioinformática.
En último lugar, para permitir la reproducibilidad de este proceso de construcción del dataset, se deja disponible el conjunto de Jupyter notebooks que sirvieron a dicho fin.
En una segunda etapa se aplicaron técnicas de aprendizaje automáticos sobre el dataset construído, lo que finalmente dio lugar a modelos de recomendación proteína-ligandos y ligando-proteínas. Si bien la performance de modelos propuestos no expuso en general buenos resultados, teniendo en cuenta el contexto de la problemática, los esquemas de validación demuestran que la capacidad predictiva de la herramienta desarrollada aún así puede ser tenida en cuenta por los profesionales del área.
Desarrollamos un pipeline a través de múltiples Jupyter notebooks, las cuales procesan distintos segmentos del circuito completo, desde el curado del dataset, hasta el entrenamiento y validación de los modelos utilizados. Esto permite que futuros desarrollos puedan mejorar los resultados generales tanto a partir de nuevas herramientas de aprendizaje automático o mediante el enriquecimiento del conjunto de datos, ambas de forma independiente, sin alterar el resto del procesamiento global.
En una tercera etapa, pudimos integrar los modelos de predicción a herramientas de Virtual Screening e Inverse Virtual Screening. En el primer caso, afortunadamente pudimos partir de un proyecto existente, LigQ, del cual nos encargamos de actualizar y migrar a una arquitectura cross-plataform gracias a la migración a Docker containers. Para el segundo, desarrollamos una nueva herramienta, Reverse LigQ, que fue desarrollada teniendo en cuenta buenas prácticas de programación, construída bajo una arquitectura modular y fácilmente modificable.
A su vez, en ambos casos proveemos toda la documentación necesaria para su puesta en marcha y dejamos el código accesible (previa autorización) en un repositorio Git. Los sistemas son de acceso público e irrestricto y se encuentran corriendo actualmente bajo las direcciones `https://ligq2.qb.fcen.uba.ar` y `https://reverse-ligq.qb.fcen.uba.ar`. Esto resulta de gran satisfacción para todas las partes involucradas en este trabajo de tesis y un valioso aporte a la comunidad científica, debido a que sus resultados pueden ser consultados sin necesidad de tener conocimientos informáticos significativos. Inclusive, en un nivel más próximo, la Plataforma Bioinformática Argentina pude dar uso inmediato a estos sistemas para enriquecer sus investigaciones actuales.

8 | Trabajo futuro

A partir del presente trabajo, se barajaron múltiples ideas de mejoras que, por cuestiones de tiempo y extensión de esta tesis, no llegaron a ser exploradas en profundidad. A continuación detallamos las más relevantes, con en el deseo de que puedan ser los disparadores de futuros trabajos dentro del área de estudio.

Confección del conjunto de datos

Dataset inicial

- Organización del dataset

- Orientar esfuerzos hacia el armado de un dataset que incluya compuestos inactivos verdaderos, es decir, que esté comprobado que no demuestran afinidad de acoplamiento con una determinada proteína objetivo. Y buscar una conformación adecuada para cada caso particular.

- Incorporar información referida a la posición relativa de los grupos aromáticos del ligando y de los aminoácidos del pocket, durante el acoplamiento.

- Incorporar información referida a la presencia de ciertos átomos (ejemplo: Nitrógeno, Oxígeno y Azufre) o de una cierta cantidad de anillos aromáticos (ejemplo: ninguno, 1, 2, 3, 4 y más de 4), codificados como valores booleanos[62]. En el segundo caso, con una codificación *one-hot*[a].

- Poder explorar el uso de nuevas features, y tener las herramientas para poder descartar de las ya utilizadas. Como se menciona en la sección 2.1, la principal razón por la cual se tomaron estas features tiene que ver con que son las que existen en la base de datos de LigQ (y a su vez provienen de ZINC). Dado que los coeficientes de correlación fuerte entre ellas no indican ninguna dependencia entre las mismas (al menos relacionándolas 1 a 1), consideramos que era un buen subconjunto de features para tomar como punto de partida. Por otro lado, las propiedades computadas de Jchem (utilizadas en trabajos previos[47]) podrían aportar nueva información a través de los cálculos que se realicen y las reglas implementadas. Sin embargo, el hecho de que sean computadas a partir de otros descriptores hace que puedan introducir cierta redundancia (al igual que las propiedades de *scoring* en pockets), lo cual haría necesario validar que no se genere ningún tipo de correlación mediante los mismos análisis estadísticos que realizamos en el capítulo 2.

- Detección de outliers

Un *outlier* se define como un valor que no forma parte de la distribución de los datos. Estos valores pueden ocasionar problemas en las técnicas de Aprendizaje Automático, dado que pueden

[a] Se trata de codificar la pertenencia a una categoría, de un total de N, como un vector de valores booleanos en donde sólo una coordenada es `True` (la categoría a la que pertenece) y las N-1 restantes son `False`.

balancear los modelos estadísticos hacia casos que no forman parte del corpus principal. Para identificarlos, `sci-kit learn` provee lo que se denomina *Local Outlier Factor*, un puntaje que mide cuán "anómala" es una muestra con respecto al dataset entero. Este valor mide la desviación local de la densidad de una muestra dada con respecto a sus vecinos. Es local porque depende de cuán aislada esté la muestra con respecto a las muestras más cercanas a ellas (sus "vecinas"). Al comparar la densidad local de una muestra con las densidades locales de sus vecinos, uno puede identificar muestras que tienen una densidad sustancialmente menor que sus vecinos. Estos son considerados valores atípicos.
Corrimos este algoritmos sobre los datos presentes en los archivos `pockets.csv` y `ligands.csv` del dataset inicial usando el valor por defecto de vecinos (parámetro del algoritmo) que `sci-kit learn` define en 20 vecinos. Los resultados arrojaron aproximadamente un 10.00% tanto para ligandos como para pockets. En este punto no profundizamos en las características de estos outliers ni en su posible efecto en los algoritmos de aprendizaje automático debido a que los métodos de escalado de features que elegimos usar durante la etapa de recomendación son robustos frente a la presencia de outliers, de manera que, en principio, éstos no significaban un problema mayor. Sin embargo, queda pendiente como trabajo futuro analizar si removiendo estos registros la performance del modelo mejora.

- PCA

Una interesante idea tenía que ver con usar PCA como features extraction, es decir, armar nuevas features a partir de las componentes obtenidas a través de PCA. Para ello creemos que es necesario profundizar en el conocimiento del área del problema, a fin de poder darle un mayor significado a cada componente.

- Estructura 3D de proteínas

- Generación de un modelo 3D (por ejemplo, mediante Modelado por Homología) de aquellas proteínas para las cuales no se pudo encontrar la estructura correspondiente en PDB.

- Identificación de pockets

- Ajuste de los parámetros de Fpocket en lugar de usar valores por defecto.
- fpocket no usa información de los ligandos presentes en el archivo PDB para identificar los pockets, algo que sí hace otra herramienta del mismo software, Dpocket. Esta recibe como parámetro el ligando presente en la proteína, y lo usa para identificar pockets "explícitos", aparte de los calculados por Fpocket.
  Un proceso de curado similar fue realizado por Wirth et al. en un paper [57] donde analizan la distribución de formas de pockets y ligandos de un conjunto de datos proveniente de sc-PDB [99] (un proyecto derivado de PDB). Utilizan un software llamado DoGSite Scorer [100] que detecta pockets, y cuenta con la posibilidad de usar un ligando presente en el archivo PDB para validar las detecciones.

Data enrichment

- Consolidación de pockets

- Considerar un umbral de similitud ϵ al comparar dos valores de una feature, en lugar del valor exacto. Gráfico variando ϵ y viendo cómo se van a agrupando los datos.
- Explorar otras métricas de distancia. Tanto para calcular este ϵ como para la clusterización.

- Estructura 3D de ligandos

- Optimización de la estructura 3D generada. En este trabajo, se usó el método `gen3D` de la aplicación Open Babel, con valores por defecto. En una primeria instancia, podemos variar los parámetros para aumentar la probabilidad de encontrar la conformación 3D que minimice globalmente la energía.
- Cabe aclarar que la conformación generada bajo el criterio anterior no necesariamente es la conformación que la molécula adopta en el acoplamiento. Un trabajo futuro podría ser enriquecer el dataset con información estructural de cada ligando considerando la proteína a la cual se acopla, en lugar de usar la misma conformación para todas las interacciones.

- Cálculo de features geométricas

- Utilizar otros métodos de aproximación de la forma de pockets y compuestos.
- En algunos casos, el algoritmo de MVEE falló a la hora de computar un elipsoide. En tales casos descartamos las observaciones. Queda pendiente estudiar las causas de dichas fallas para recalcular el elipsoide.

Extensión del dataset usando datos de ChEMBL

- Bolsillos

Al tomar los datos de ChEMBL, podemos implementar reglas ad-hoc para elegir mejor el pocket en el que creemos que se dio la unión, en lugar de tomar el de mejor puntaje. Por ejemplo, podemos analizar la carga que tiene, y decidir en función de eso si la unión sería efectivamente factible o no. El mismo razonamiento puede aplicarse a características como volumen. Cabe recordar que el score es una *feature* computada sobre características del mismo bolsillo, y no tiene en cuenta ninguna característica del ligando ni de la interacción con éste. Es probable incluso que si tomamos varios pockets de la misma molécula, muchos tengan puntajes ligeramente distintos, con lo cual puede ocurrir que el "mejor" sólo sea ligeramente mejor que los siguientes en el orden.

- Ligandos

Una de las propiedades moleculares que decidimos descartar del dataset inicial PDB al momento de extenderlo con datos provenientes de ChEMBL fue la carga neta de la molécula (campo `charge`), dado que no estaba disponible en esta última. Sin embargo, puede ser calculada a partir de la composición molecular en una próxima etapa de enriquecimiento del dataset.

- Interacciones

Con respecto a las interacciones, en la FAQ de ChEMBL[101] aclara que puntúa la interacción con números del 1 al 9, los cuales representan lo siguiente:

> Como parte del proceso de curación manual de los datos, se asigna un *puntaje de confianza* a la relación ensayo-molécula objetivo existentes en la base de datos. El puntaje de confianza refleja tanto el tipo de molécula objetivo asignada a un ensayo particular como la confianza de que la molécula objetivo sea la correcta para ese ensayo. El puntaje de confianza varía entre 0 (para las entradas aún no curadas) a 9, en donde a una única proteína se le asignó un alto nivel de confianza. Ensayos con objetivos no-moleculares o un organismo, reciben un nivel de confianza igual a 1, mientras que ensayos con objetivos proteicos reciben un puntaje de confianza de al menos 4 puntos. Se detalla a continuación la descripción de los puntajes (en inglés):.
>
> 0 Default value - Target assignment has yet to be curated
>
> 1 Target assigned is non-molecular
>
> 3 Target assigned is molecular non-protein target
>
> 4 Multiple homologous protein targets may be assigned
>
> 5 Multiple direct protein targets may be assigned
>
> 6 Homologous protein complex subunits assigned
>
> 7 Direct protein complex subunits assigned
>
> 8 Homologous single protein target assigned
>
> 9 Direct single protein target assigned

Por cuestiones de falta de tiempo, el BIA nos proporcionó los datos sin aplicar un filtro en base a estos puntajes, pero cabe pensar que si una vez realizado la cantidad de datos resulta estar por encima de un cierto umbral, bien puede tener sentido probar la performance de nuestro modelo con estas variantes.

Predicción proteína-ligandos

Opciones de configuración de predictores

Existen múltiples métricas de distancia. Se propone realizar nuevas pruebas del NN Regressor utilizando otra métrica. La distancia euclediana, la distancias de Chebyshev, de Minkowski, de Wminkowski o de Mahalanobis son algunas de ellas[b].

Predicción del ligando ideal

- Elección del modelo de regresión

Una de las desventajas de usar una Red Neuronal como modelo de regresión es que funciona como una "caja negra". Sabemos que recibe un vector de *features* de un pocket real y produce

[b] El listado completo de las métricas aceptadas por la clase `KNNRegressor` puede encontrarse en la documentación correspondiente a Distance Metrics de `sci-kit learn`: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html`.

como salida un vector de *features* de un ligando ideal, pero no contamos con un conjunto de reglas explícitas para entender cómo se llega a ese resultado. La toma de decisiones se encuentra codificada en su arquitectura (dadas por la secuencia de matrices de peso y funciones de activación de las capas). Como trabajo futuro, se puede considerar una arquitectura que sea "explicable" (ver sección 8 para una definición en el contexto de Inteligencia Artificial) al momento de definir el modelo.

- Feature importance sobre la salida

Si bien la técnica de *feature importance* brinda información sobre las features de entrada del modelo, podría ser interesante conocer la predictibilidad de las features de salida: quizás sea conveniente tener un buen modelo que estime correctamente un subconjunto de características de la salida antes que un modelo que estime erróneamente el total de ellas.

"Explicabilidad" del modelo

La **explicabilidad** (en inglés, *explainability*) de un modelo de aprendizaje automático se refiere al grado en que la mecánica interna de un algoritmo de *deep learning* o de aprendizaje automático puede ser explicado en términos humanos. Es decir, un experto humano debería poder conocer cómo y por qué un modelo ha arribado a una determinada predicción, y darle sentido a los datos en el contexto del dominio de aplicación[102].
Hay muchos argumentos a favor de explicabilidad en sistemas de Inteligencia Artificial (IA). Los principales son[103]:

- Verificación del sistema (por parte de expertos)
- Mejora del sistema: detectando falencias y sesgos, selección de modelos.
- Aprender del sistema: *"un dominio donde la extracción de información a partir del modelo puede ser crucial son las ciencias. La gente de ciencia está más interesada en identificar las leyes ocultas de la naturaleza que en predecir cantidades de un modelo de caja negra"*
- Conformidad con la legislación: implementación de un derecho a la explicación, para conocer el por qué de una decisión algorítmica.

Los avances recientes en metodologías de *deep learning* han hecho que los sistemas de IA hayan alcanzado la capacidad de los seres humanos en realizar tareas complejas. Sin embargo, el área de "IA Explicable" aún es relativamente nueva, y por ahora los modelos más explicativos tienden a sacrificar performance. Creemos que sumar estas herramientas a los procesos de *Virtual Screening* e *Inverse Virtual Screening* podría dar un salto en cuanto a la comprensión de los modelos utilizados hasta el momento y orientar futuros experimentos.

LigQ y reverse-LigQ

Novelty Detection

Dado el extenso análisis que realizamos sobre el conjunto de datos, podríamos brindar al usuario mayor información con respecto a las características del pocket/ligando que está introduciendo como input en cualquiera de los dos sistemas. Un concepto interesante en este aspecto es el

de detección de **novedades** (en inglés, Novelty Detection[104]). A diferencia de la detección de *outliers*, que busca valores extraños en el dataset existente, la detección de novedades busca decidir si una *nueva* observación es o no un outlier del dataset. Conociendo la distribución de los datos, sería interesante informar al usuario acerca de si el dato que está introduciendo es anómalo (e.g. un ligando con un volumen muy grande) y de esta forma dar algún resguardo acerca de la predicción brindada por el sistema de recomendación.

Predicción ligando-proteínas

- Extender el conjunto de datos con pockets de proteínas sin interacciones

Al finalizar el pipeline de recomendación proteína-ligandos se proponen compuestos similares a un ligando ideal, que no necesariamente tienen una entrada en la tabla de acoplamientos. Sin embargo, en el pipeline de recomendación ligando-proteínas todos los pockets similares al pocket ideal pertenecen a una proteína que participó de al menos una interacción con un ligando. Esto se debe, en primer lugar, a que sólo se usaron proteínas que tuviesen una estructura en PDB y, en segundo lugar, a que el conjunto de datos se enfocaba fuertemente en casos positivos de interacción proteína-ligandos, dejando de lado aquellas sin información experimental.
Es de vital importancia buscar posibles usos de un ligando no sólo en proteínas experimentalmente testeadas, sino también en aquellas para las que se desconoce ligandos que se acoplen a estas o, incluso sin función conocida. Para extender la cantidad de pockets sobre los cuales realizar la búsqueda hay que incorporar proteínas excluidas del dataset, ya sea por los filtros del dataset inicial o por no contar con estructura 3D. En el segundo caso, se podrían generar modelos por homología.

Implementación del software reverse-LigQ

- Incorporar recomendación de proteínas por similitud con una proteína objetivo existente

De la misma manera que LigQ implementa un filtro colaborativo para la recomendación proteína-ligandos, el pipeline implementado por reverse-LigQ podría sumar a su set de herramientas filtros relacionados a similitud item-item. Un ejemplo de esto sería utilizar métricas como la distancia de Tanimoto.

- Filtrado de ligandos en base a su usabilidad

Como se mencionó en el capítulo 1, la disposición de un fármaco en el cuerpo depende de 4 factores denominados ADME. Existe una regla empírica conocida como *Regla de Lipinski* que permite estimar qué tipo de sustancias químicas pueden funcionar como buenas drogas de consumo oral en humanos. La regla fue enunciada por Christopher A. Lipinski en 1997[105], observando que la mayoría de los compuestos químicos utilizados en medicamentos son moléculas relativamente pequeñas y lipofílicas (que presentan una afinidad con compuestos lipídicos). Este conjunto de reglas se basa en las características de importancia farmacocinética en el cuerpo humano, principalmente `ADME`, pero no intenta predecir si un compuesto será farmacológicamente activo o no. La regla dice que, en general, un principio activo administrable por vía oral no debe violar más de una de las siguientes consideraciones:

- No debe contener más de cinco donantes de enlaces por puente de hidrógeno.
- No debe contener más de diez receptores de enlaces por puente de hidrógeno.
- Debe poseer un peso molecular inferior a las 500 uma.
- Debe tener un coeficiente octanol-agua (log P) inferior a 5,3.

Esta reglas es conocida como *Regla de Cinco de Lipinski* dado que todos los valores involucrados son múltiplos de 5.
Una interesante alternativa sería introducir esta validación dentro del pipeline de reverse-LigQ, a fin de informar al usuario acerca de qué tan útil resulta el ligando objetivo que está analizando.

- Carga manual del archivo SDF

El backend recibe en la consulta el ID del compuesto y lo utiliza para descargar el archivo SDF correspondiente desde RCSB PDB. Si bien el proceso ocurre sin intervención del usuario, cuenta con la limitación de que RCSB ofrece sólo dos variantes de la estructura molecular. A futuro, el sistema podría permitir al usuario cargar un archivo SDF del compuesto en una conformación espacial específica.
Una consecuencia de esta mejora es que el sistema no estará limitado a los compuestos presentes en PDB para generar los descriptores geométricos.

9 | Notas finales

Lecciones aprendidas

- La *Ciencia de Datos* se trata de un proceso iterativo de aprendizaje. Los datos siempre pueden refinarse aún más a medida que vamos comprendiendo el dominio del problema. Surgen nuevas preguntas que nos hacen ver los datos desde otra perspectiva. La revisión de los modelos es periódica (ver figura 9.1).

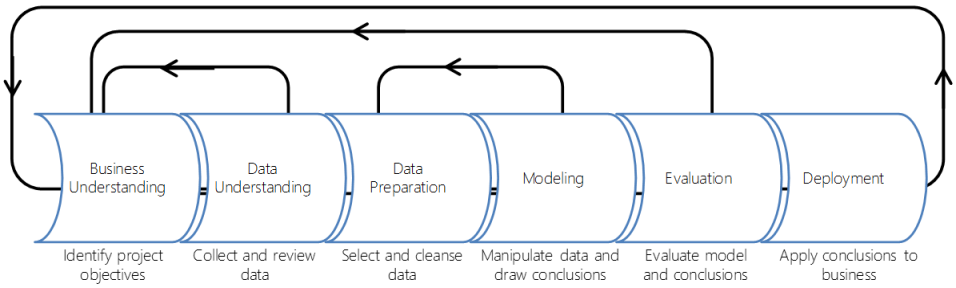


Figure 9.1: *Cross Industry Standard Process for Data Mining* (CRISP-DM).

- Al comenzar nuestro planteo por sobre las múltiples bases de datos, chequeamos que la información que nos llegaba procesada sea coherente con lo que figuraba en la plataforma web, debido a que los datos se encuentran siendo constantemente actualizados[a]. Validaciones de este tipo resultaron ser muy útiles no sólo para mejorar los datos con los que contábamos sino también para adquirir conocimiento del dominio del problema.

- Algo que ciertamente aprendimos fue la importancia de plantear desde el primer momento el enfoque con el que encararíamos el trabajo, lo cual en cierto punto excede el campo de estudio en sí mismo. Al trabajar sobre desarrollos previos, dos preguntas aparecieron de forma clara desde el inicio: *¿qué tenemos?* y *¿cómo podemos mejorarlo?*. Y más interesante aún, cómo distribuiríamos el tiempo en responder cada una. Lo cierto es que no seguimos ninguna recomendación en particular, sino que se fue dando de manera natural. Finalmente, de forma aproximada, podemos concluir que alrededor de 60% del tiempo de este trabajo fue destinado a comprender el dominio del problema, los desarrollos anteriores y, por sobre todo, el dataset con el que trabajams. El restante 40% estuvo mas bien destinado a: implementar los modelos de aprendizaje automático; desarrollar las piezas de software (LigQ y reverseLigQ) que se valen de dichos modelos para dar resultados útiles a la comunidad científica y médica del mundo; y por último a comunicar todo este trabajo de la forma más clara y amena posible mediante el presente informe.

[a] Por poner un ejemplo, en los datos de ChEMBL que recibimos en primera instancia, la propiedad `logp` figuraba como 0 para muchos registros, mientras que en la versión web de ChEMBL ya figuraba con valores definidos. Por este motivo procedimos a realizar una actualización con la última versión de ChEMBL, lo cual solucionó el inconveniente

- Finalmente, pudimos plasmar todo este aprendizaje y exploración en el presente informe, el cual conllevó una gran cantidad de revisiones y reescrituras, pero que definitivamente creemos que puede resultar de mucha utilidad para futuros estudiantes que quieran profundizar en la temática de *Virtual Screening* e *Inverse Virtual Screening*.

A | Apéndice: métodos y materiales

A.1 Introducción

Esta sección comprende fundamentos de bioinformática relevantes para entender mejor el dominio del problema como la estructura proteica y las técnicas que se utilizan para calcular su representación tridimensional.

A.1.1 Determinación de la estructura tridimensional

A continuación detallamos los tres enfoques principales con los cuales se logra obtener una representación 3D de una proteína: la determinación experimental, el modelado comparativo y la predicción *ab initio*.

A.1.1.1 Determinación experimental

Los métodos experimentales más difundidos son[106]:

- **Cristalografía de Rayos X:** la proteína es purificada y cristalizada, y luego sometida a un intenso haz de rayos X. Las proteínas en el cristal difractan el haz en uno o varios patrones de puntos característicos que son luego analizados para determinar la distribución de los electrones en la proteína. El mapa de densidad de electrones es entonces interpretado para determinar la ubicación de cada átomo. El proceso completo puede observarse en la Figure A.1.
- **Espectroscopía de Resonancia Magnética Nuclear (RMN):** la proteína es purificada, ubicada en un fuerte campo magnético, y luego explorada con ondas de radio. Del análisis de las mediciones obtenidas se obtiene una lista de condiciones acerca de qué núcleos atómicos deben estar cerca y cómo debe ser la conformación local. Luego, a partir de esta información se construye un modelo de la proteína que muestra la ubicación de cada átomo (aunque usualmente se compone de un ensamblado de modelos, todos consistentes con la lista de restricciones observadas). A diferencia de la técnica de cristalografía, la proteína se analiza en solución, con lo cual es más útil a la hora de estudiar la estructura de proteínas flexibles.

En la mayoría de los casos, la información experimental no es suficiente para construir el modelo atómico. Se suele agregar al modelo el conocimiento previo, como la estructura molecular, la secuencia de aminoácidos y la geometría preferida de los átomos (distancia de enlaces y ángulos).

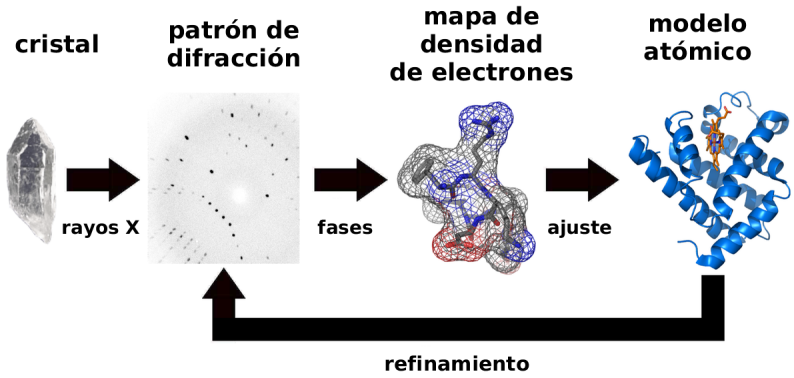


Figure A.1: Diagrama de flujo para la determinación de la estructura molecular por cristalografía de rayos X. *Fuente: Adaptado de Thomas Splettstoesser (CC BY-SA 3.0 Wikimedia Commons)*

Finalmente, como fase final del proceso, las estructuras tridimensionales suelen ser incorporadas a PDB, que aloja información estructural obtenida experimentalmente[a], no sólo de proteínas sino también (en menor medida) de otras moléculas biológicas de gran tamaño, como lo son los ácidos nucleicos.

A.1.1.2 Modelado comparativo (basado en homología)

No todas las secuencias conocidas tienen su correspondiente estructura 3D resgistrada en PDB. Esto se debe a limitaciones experimentales. Algunas proteínas son difíciles de producir o purificar en las cantidades necesarias. Otras no cristalizan lo suficientemente bien para el uso de rayos X, que necesita muchas moléculas alineadas y orientadas de la misma forma. La espectroscopia de RMN, por otro lado, está limitada a proteínas pequeñas o de tamaño mediano.
Las proteínas de membrana, especialmente las GPCRs, son un buen ejemplo de familias de proteínas de gran valor biológico (como blanco de drogas) de las cuales sólo se ha podido obtener un pequeño número de estructuras, debido a la dificultad de extraerlas de la membrana celular, por su flexibilidad e inestabilidad[108]. Éstas constituyen menos del 2% de las estructuras proteicas registradas en PDB[107,109].
Una de las técnicas desarrolladas para hacer frente a estas limitaciones es el modelado por homología. Esta intenta predecir la estructura terciaria de una proteína (*target*) a partir de una o más proteínas relacionadas (homólogas) con estructura conocida (*templates*). Como primer paso, se selecciona de una base de datos aquellas proteínas con similitudes detectables y con las que pueda construirse un alineamiento correcto de secuencias. Luego continúa la etapa de construcción del modelo, que consiste en transferir información de la estructura ya conocida (matcheo de segmentos, restricciones espaciales, etc.) a la secuencia target, como puede visualizarse en la Figure A.2.
Finalmente, se refina el modelo y es común que a continuación siga una etapa de validación de la calidad del modelo para una determinada aplicación. En caso de no ser apropiado, debe ser descartado y empezar nuevamente con otro template.

[a] Un dato interesante es que del 90% de las estructuras proteicas de PDB han sido determinada mediante cristalografía de rayos X[107].

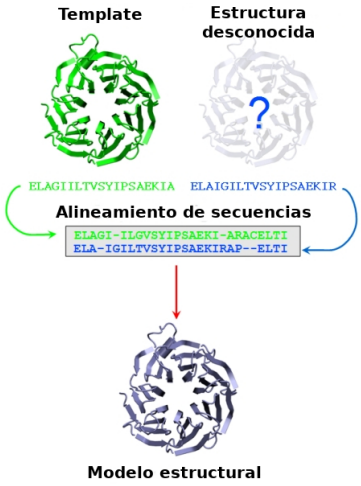


Figure A.2: Modelado por homología *Fuente: https://www.unil.ch/pmf*

Entre los programas de computadora existentes en la actualidad para la etapa de generación de modelos por homología podemos citar *Modeller*[110], el cual es utilizado por LigQ cuando no encuentra el modelo correspondiente de la proteína objetivo (típicamente, si sólo se conoce su secuencia proteica) en PDB. Para más detalle de este proceso, consultar el chapter 4.

Limitaciones

El modelado por homología se basa fuertemente en la hipótesis de que un pequeño cambio en la secuencia de una proteína resulta sólo en un pequeño cambio en su estructura 3D[111], por lo tanto es importante elegir proteínas con similitud de secuencia suficientemente alta. El porcentaje mínimo de residuos similares depende de la longitud del alineamiento[6] (ver Figure A.3). Como guía se considera que, por debajo de 20% - 30%, las proteínas no son homólogas y, en consecuencia, el modelo generado no será correcto[112]. Por encima de este umbral, la probabilidad de que las proteínas sean homólogas es alta. La exactitud buscada del modelo dependerá del contexto de su uso.

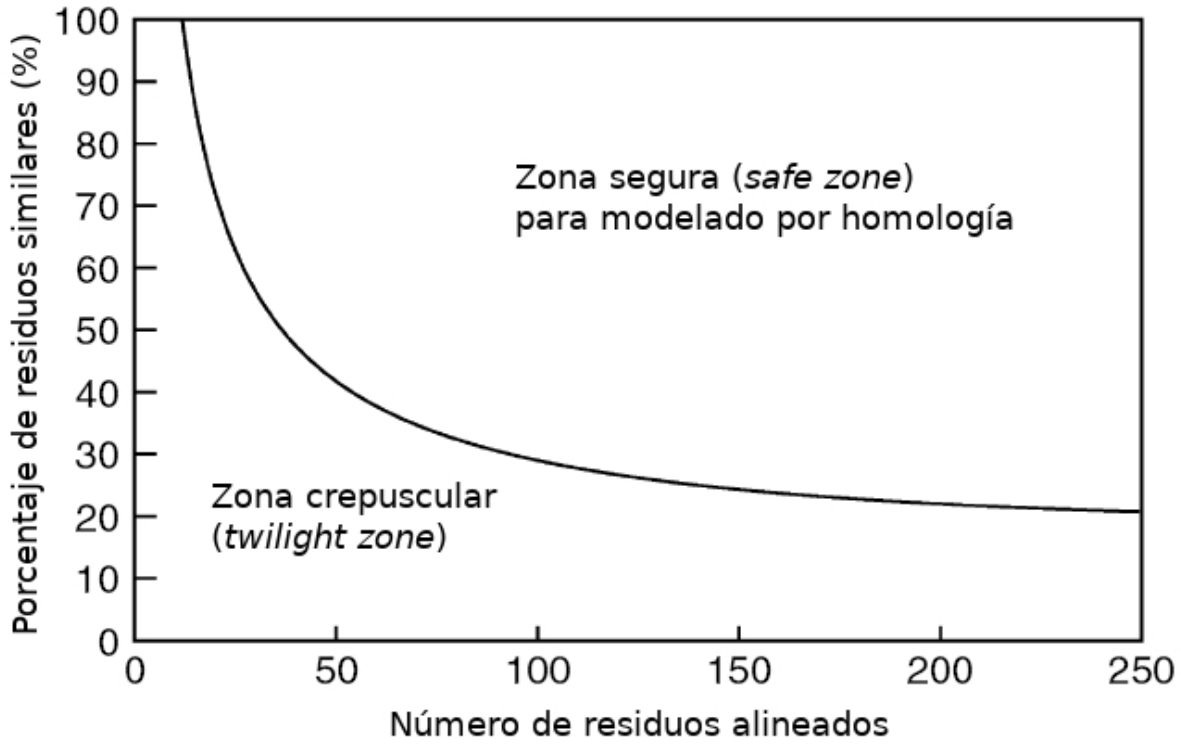


Figure A.3: Curva que establece el umbral de homología entre dos proteínas a partir de su secuencia y longitud del alineamiento: en la "zona segura" la estructura es confiable para ser usada como template. En la "zona crepuscular", el alineamiento no proporciona suficiente información sobre similitud de estructura*Fuente: adaptado de Krieger (2003)* [113]

.

A.1.1.3 Predicción *ab initio* (*de novo*)

Debido a las limitaciones de los métodos experimentales y a la ausencia de modelos apropiados para la utilización de modelado comparativo, se desconoce la estructura 3D de gran parte de las secuencias proteicas conocidas hasta el momento.
Para dar una idea de la magnitud del problema, a enero de 2017[114], de las 14.849 familias de proteínas de la base de datos PFam (ver subsection A.2.1):

- sólo 4.752 tenían al menos un miembro con estructura determinada experimentalmente por cristalografía de rayos X o espectroscopía de RMN.
- para 3.984 podía obtenerse un modelo confiable a partir de estructuras homólogas de estructura conocida usando el software `HHsearch`. Y, adicionalmente, para otras 902 podía construirse modelos menos confiables.
- las restantes 5.211 familias (poco más de un tercio) no tienen asociada información estructural.

En el caso de no contar con información estructural, se recurre a métodos de predicción *ab initio*, que intentan generar una estructura "desde cero". Partiendo desde la estructura primaria de la proteína y modelando la energética del proceso de plegado, estos algoritmos tienen como

objetivo llegar a una estructura que minimice la energía libre, lo cual no es una tarea trivial. De hecho, los problemas: a) qué balance de fuerzas interatómicas dicta la estructura de la proteína a partir de los aminoácidos; b) cómo ocurre el proceso de plegado; y c) el problema computacional de predecir la estructura nativa de una proteína (i.e. la más estable) a partir de su secuencia de aminoácidos, han sido unos de los grandes retos de la biología molecular del último medio siglo. Colectivamente se conoce como "el problema del plegamiento de proteínas" (en inglés, *the protein folding problem*)[115]. Por fortuna, avances en las técnicas *ab initio* recientes han permitido generar modelos 3D de más de 600 familias de proteínas, que se estima, abarcan millones de proteínas[114].

A.2 Dataset

Aquí se detalla todo lo relacionado a la generación del *datasets*: las bases de datos biológicas consultadas, los mecanismos para obtener los *pockets* de una proteína y la obtención de *features* de éstos últimos.

A.2.1 Bases de datos biológicas

Las bases de datos de acceso público a las cuales hemos recurrido directa o indirectamente (a través de los datasets de partida) son:

- **UniProt**[116]: repositorio de secuencias proteicas. Secuencias de proteínas y sus anotaciones. Las secuencias se encuentran almacenadas en formato FASTA.
- **PDB**[117]: repositorio de estructuras tridimensionales de proteínas y ácidos nucleicos, obtenidos experimentalmente; también contiene estructuras de compuestos tipo droga. El instituto RCSB (*Research Collaboratory for Structural Bioinformatics*) es responsable de administrar el datacenter para PDB ubicado en Estados Unidos.
- **ChEMBL**[63]: una base de datos curada manualmente, de moléculas bioactivas con propiedades tipo-droga, e interacciones. En el Apéndice A, sección **ChEMBL** se detallan más precisiones al respecto.
- **Pfam**[118]: base de datos de familias de proteínas, representadas por alineamiento múltiple de secuencias y modelos ocultos de Markov (*HMM*)
- **ZINC**[119]: base de datos de compuestos comercialmente disponibles, para *virtual screening*. LigQ posee una copia local de esta base de datos, en base a la cual ofrece resultados al usuario. En el Apéndice A, sección **ZINC**, se encuentra explicado con detalle la información que contiene en esta base de datos.
- **UniChem**[120]: base de datos que establece mapeo de identificadores moleculares entre distintos repositorios.

A.2.2 Formatos de archivos

Existen distintos formatos estándar de archivos para el almacenamiento e intercambio de información química y biológica (ej: identificación, composición química, estructura molecular). En la tabla Table A.1 se detallan aquellos que utilizamos a lo largo de este trabajo).

Formato	Usos
SMILES	Notación lineal para describir la estructura de sustancias químicas usando cadenas ASCII.
InChi	Identificación de sustancias químicas y codificación de información molecular, para facilitar búsquedas en bases de datos. Propuesto por IUPAC como estándar.
FASTA	Representación lineal de secuencias de ácidos nucleicos y de aminoácidos, usando una letra para representar cada nucleótido o aminoácido, respectivamente.
PDB	Información estructural tridimensional de proteínas. Aunque en menor medida se usa para otros compuestos (ligandos, cofactores, cosolventes, etc).
SDF	Información estructural de moléculas. Puede codificar la información de más de una molécula en el mismo archivo.
PQR	Formato de fpocket para codificar la estructura del pocket de modo análogo al formato PDB, en función de las α-esferas y sus radios, en lugar de átomos.

Table A.1: Formatos de archivo de información química y biológica

A.2.3 ChEMBL

ChEMBL consiste en un conjunto de fuentes de datos pertenecientes a literatura científica, patentes, conjuntos de datos depositados, PubChem BioAssay and BindingDB databases, conjuntos de datos de toxicología y drogas. La lista actual de fuentes se lista en la tabla A.2[b].

[b] Toda esta información se encuentra descripta en el FAQ del sitio web oficial de ChEMBL: `https://chembl.gitbook.io/chembl-interface-documentation/frequently-asked-questions`.

1. Scientific Literature
2. GSK Malaria Screening
3. Novartis Malaria Screening
4. St Jude Malaria Screening
5. Sanger Institute Genomics of Drug Sensitivity in Cancer
6. PubChem BioAssays
7. Clinical Candidates
8. Orange Book
9. Open TG-GATEs
10. Manually Added Drugs
11. USP Dictionary of USAN and International Drug Names
12. Drugs for Neglected Diseases Initiative (DNDi)
13. DrugMatrix
14. GSK Published Kinase Inhibitor Set
15. MMV Malaria Box
16. TP-search Transporter Database
17. Harvard Malaria Screening
18. WHO-TDR Malaria Screening
19. Deposited Supplementary Bioactivity Data
20. GSK Tuberculosis Screening
21. Open Source Malaria Screening
22. External Project Compounds
23. Gene Expression Atlas Compounds
24. AstraZeneca Deposited Data
25. FDA Approval Packages
26. GSK Kinetoplastid Screening
27. K4DD Project
28. Curated Drug Metabolism Pathways
29. St Jude Leishmania Screening
30. Gates Library compound collection
31. MMV Pathogen Box
32. HeCaToS Compounds
33. Withdrawn Drugs
34. BindingDB Database
35. Patent Bioactivity Data
36. Curated Drug Pharmacokinetic Data
37. CO-ADD antimicrobial screening data
38. WHO Anatomical Therapeutic Chemical Classification
39. British National Formulary
40. Published Kinase Inhibitor Set 2

Table A.2: Lista de las fuentes que conforman ChEMBL.

A.2.4 Construcción del dataset original

Cómo se aclaró anteriormente, la construcción del dataset original fue tarea desarrollada por el BIA. Las siguientes instrucciones detalladas fueron corridas sobre un entorno de desarrollo GNU/Linux. **Los archivos se encuentran bajo propiedad del BIA, para obtenerlos contactarse con ellos.**

1. Descargar la versión más reciente de PDB. Para esto, se descarga y edita el script de sincronización `ftp` de PDB:

   a Instalar `wget` ejecutando `apt-get install wget`

   b Ejecutar `wget http://ftp.wwpdb.org/pub/pdb/software/rsyncPDB.sh`

   c Editar el archivo `rsyncPDB.sh` en las líneas:

   ```
   MIRRORDIR=<your local ftp directory>   # your top level rsync directory
   LOGFILE=<your local ftp directory>/logs # file for storing logs
   RSYNC=<your local>/rsync               # location of local rsync
   ```

   Completar con los datos locales. Por ejemplo:

   ```
   MIRRORDIR=/data/pdb
   LOGFILE=pdb_log
   RSYNC=/usr/bin/rsync
   ```

   d Descomentar las siguientes líneas:

   ```
   #PORT=873 # port PDBe server is using
   #SERVER=rsync.ebi.ac.uk::pub/databases/rcsb/pdb-remediated # PDBe server
       name
   #${RSYNC} -rlpt -v -z --delete --port=$PORT ${SERVER}/data/structures/
       divided/pdb/ $MIRRORDIR > $LOGFILE 2>/dev/null
   ```

e Otorgar permisos de ejecución al script ejecutando `chmod +x rsyncPDB.sh`

f Instalar `rsync` ejecutando `sudo apt-get install rsync`

g Ejecutar `./rsyncPDB.sh`. Esto creará múltiples subdirectorios con los diferentes pdbs ordenados *(nota: puede tardar varias horas en correr).*

2. Para elminiar cierta redundancia producto del sesgo experimental (proteínas que aparecen muy frecuentemente), se procede a realizar un clustering por identidad $\geq 100\%$:

   a Descargar el archivo `fasta` de todas las proteínas de pdb ejecutando `wget http://ftp.wwpdb.org/pu`

   b Instalar `cd-hit` ejecutando `sudo apt-cache search cd-hit`

   c Ejecutar `cd-hit -i pdb_seqres.txt -c 1.0 -o pdb_seqres.id100.fasta`

3. Ejecutar `batch_findPockets.py -l pdb_seqres.id100.lista` para correr la tarea de LigQ en batch sobre la lista de pdbs seleccionados *(nota: es el paso más largo de todos, puede tardar incluso semanas).*

4. Crear y poblar la base de datos ejecutando
   `load_features_sql.py -l pdb_seqres.id100.lista -d ''protlig''`

5. Filtrar ligandos espurios ejecutando (sólo se mantienen los que corresponden a ligandos tipo droga): `filter_sql.py -d ''protlig'' -f filter_file.txt`. Esto da como resultado 3 archivos: `proteins.csv`, `ligands.csv` y `prot_lig.csv`.
   (nota: adicionalmente pueden incluirse otros filtros)

6. Filtrar proteínas por identidad usando clusterización por identidad $\geq 90\%$:

   a Filtrar las entradas de `fasta` con la lista de proteínas del punto anterior ejecutando `filter_fasta.py -i pdb_seqres.id100.fasta -l proteins.csv`

   b Ejecutar `cd-hit` para el archivo `fasta` de pdb previamente filtrado ejecutando `cd-hit -i pdb_seq`

7. Agrupar en por familias y selección de 3 proteínas por familia como máximo:

   a Instalar hmmer3 ejecutando `sudo apt-get install hmmer3`

   b Ejecutar `group_by_pfam.py -f pdb_seqres.id90.fasta`. Los archivos de salida son `pdb_seqres.id90.pfam3.fasta` y `pdb_seqres.id90.pfam3.lista`.

   c Filtrar archivo original con resultados de la agrupación de familias ejecutando: `filter_csv.py -i p`
   El archivo de salida es `proteins.filter.csv`.

Este proceso completo puede visualizarse gráficamente en la figura A.4.

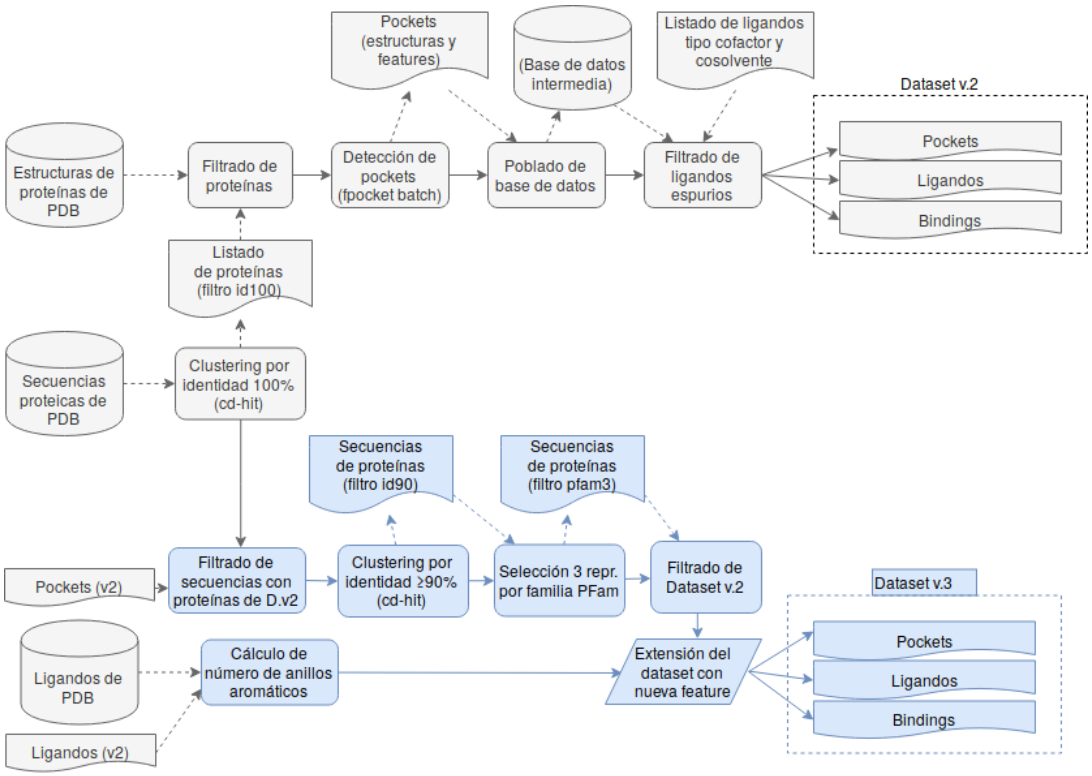


Figure A.4: Etapas principales de la generación del dataset de partida de esta tesis.

A.2.5 Propiedades (*features*)

A.2.5.1 Detalle de propiedades de los ligandos

- `MWT`: peso molecular del ligando, calculado como la suma de todos los pesos atómicos de los elementos que lo componen, multiplicado por la cantidad de átomos de ese elemento en la fórmula molecular.
- `LogP`: logaritmo del coeficiente de partición del ligando. El coeficiente de partición es el cociente entre las concentraciones de una sustancia en las dos fases de una mezcla formada por dos solventes inmiscibles en equilibrio. El coeficiente mide la solubilidad diferencial de una sustancia en esos dos solventes. Los solventes más usuales son agua y un solvente hidrófobo como el octanol, aportando información de la lipofilicidad de una sustancia.
- `LogS`: logaritmo de la solubilidad del ligando en agua. La solubilidad de un compuesto se define como la concentración de ese compuesto en una solución saturada, cuando ésta se encuentra en equilibrio con la fase sólida de ese compuesto.
- `Desolv_apolar`: energía de solvatación en solvente no polar (kcal/mol).
- `Desolv_polar`: energía de solvatación en solvente polar (kcal/mol).
- `HBD`: cantidad de donantes de puentes de hidrógeno. Un donante de puente de hidrógeno es el átomo que está unido covalentemente al hidrógeno participante del puente de hidrógeno. Suele ser un átomo muy electronegativo, como N, O o F. La cantidad de donantes es una

medida de la habilidad de la molécula para formar puentes de hidrógeno como donante. Se calcula sumando los hidrógenos unidos a oxígenos o nitrógenos que no tengan carga negativa.

- `HBA`: cantidad de aceptores de puentes de hidrógeno. Un aceptor de puente de hidrógeno suele ser un átomo muy electronegativo con un par de electrones libres (N, O, F). La cantidad de aceptores es una medida de la habilidad de la molécula para formar puentes de hidrógeno como aceptor. Se calcula sumando los átomos de oxígeno, nitrógeno y flúor, excluyendo los nitrógenos con carga positiva, alto estado de oxidación o pirrólicos.
- `tPSA`: *topological polar surface area*, es definida como la suma de las superficies de átomos polares (usualmente oxígenos, nitrógenos e hidrógenos) en una molécula, en $Å^2$
- `Charge`: la carga neta de la molécula.
- `NRB`: *number of rotatable bonds*, es la cantidad de enlaces que permiten la libre rotación alrededor de sí mismos. Se lo define como cualquier enlace simple, no en un anillo, ligado a un átomo pesado no terminal. Excluyendo del conteo los enlaces amida C- N, debido a su alta barrera de energía rotacional[121].

A.2.5.2 Valores imputados

En el caso de `Desolv_apolar` y `Desolv_polar`, de un total de 122.095, 23.221 tenían valor NAN, mientras que los restantes 98874 tenían valor 0. Al estar obligados a descartar los NaN, queda un único valor restante, lo cual hace que no aporte ningún tipo de valor para la construcción de modelos de recomendación. Es una propiedad con entropía nula. En el caso de `LogS` el caso es ligeramente distinto, 6% de los valores son NaN, 86% tienen valor 0 y el restante 4% tiene valores diferentes. Nuevamente, debido a que la distribución es prácticamente un único valor, decidimos no tomar esta propiedad en los futuros análisis.

A.2.6 fpocket

Fpocket puede detectar diferentes tipos de bolsillos en función de los parámetros con que se lo configure. La herramienta fue creada inicialmente para detectar pequeños sitios de unión de moléculas en proteínas. Eso es en lo que los la mayoría de los casos se busca (o al menos los desarrolladores así lo asumieron)[40]. Pero como se quiere abarcar a un número máximo de casos, fpocket intenta mantenerse lo más flexible posible a través diversos parámetros. Por ejemplo, estas configuraciones pueden usarse para la detección de canales y vías de gas en una proteína, lo cual es un tema completamente diferente en comparación con la búsqueda de unión a fármacos.

A.2.7 Determinación de *pockets* de una proteína

Una vez determinada la estructura de la proteína, es importante determinar cuáles son las regiones de la misma en donde puede producirse un acoplamiento con un compuesto, es decir las **cavidades** o *pockets*. Éstos poseen ciertas propiedades (volumen, polaridad, etc.) que lo hacen apto para que una molécula pequeña pueda ingresar en el y unirse al mismo (ver Figure A.5). Este acoplamiento, en algunos casos, termina inhibiendo o modulando la función de la proteína como tal, en cuyo caso nos referimos a dicha cavidad como *sitio activo*.

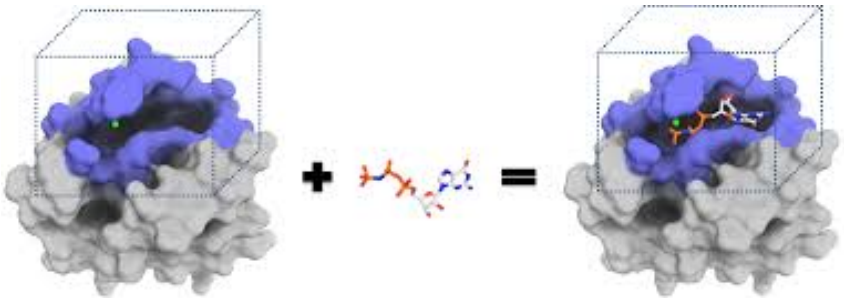


Figure A.5: El bolsillo es una región en la estructura molecular apta para que un compuesto pueda acoplarse. Imagen obtenida de *AutoDock: Computational docking of small molecules for drug discovery* (`https://en-lifesci.tau.ac.il/bioinformatics-unit/Autodock_2016`)

La proporción de estructuras cristalinas en las que la ubicación del sitio activo está determinada es bajo, y no es posible hacer extrapolaciones a modelos de manera directa. Es por ello que para nuestro trabajo decidimos utilizar directamente algoritmos de detección de bolsillos y luego evaluar las características de éstos.
Si bien existen diferentes estrategias para encontrar cavidades en la estructura proteica, los métodos existentes no relacionan esta información con las propiedades que debiera tener un compuesto para unirse a la cavidad encontrada. **Uno de los objetivos de esta tesis es crear esta asociación**. Después de evaluar diferentes estrategias para resolver este problema, hemos elegido para las distintas etapas de esta tesis el algoritmo **fpocket**[40], el cual está basado en una estrategia que utiliza α-spheres y la teselación de Voronoi (ver Figure A.6). **Fpocket** es, además, la herramienta de detección de bolsillos utilizado por LigQ.
Simplificadamente, una α-sphere es una esfera que contacta cuatro átomos de la estructura de la proteína en sus límites y que no contiene (encierra) átomos dentro de la misma. Los cuatro átomos, por definición, se encuentran a la misma distancia del centro de la esfera (su radio). Para el caso de las proteínas, pequeñas esferas pueden ser ubicadas dentro de la estructura, y esferas de radio mayor pueden ser ubicadas en su superficie. Los valores permitidos para el radio de las esferas son un parámetro del algoritmo. Valores más pequeños se utilizan, por ejemplo, para detectar túneles en la estructura; cuatro átomos en un plano definen técnicamente una esfera de radio infinito; los tamaños de esfera óptimos para detectar cavidades que alojan ligandos tienen un rango definido y validado empíricamente[40].

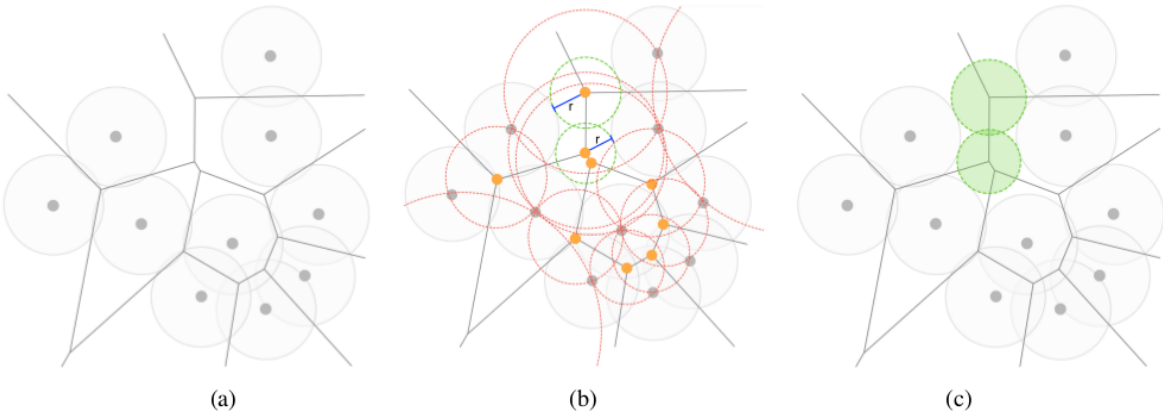


Figure A.6: Detectando cavidades mediante **fpocket**: (a) diagrama de Voronoi de los centros atómicos; (b) al igual que una bola de Voronoi (círculos punteados de color rojo), cada *α-sphere* (círculo punteados de color verde) también está centrada en un vértice de Voronoi (puntos naranjas), pero es una esfera de contacto que es tangencial a la superficie de los átomos (círculos rellenos de color gris); (c) grupo de *α-spheres* (círculos rellenos de color verde) que rellenan la cavidad.[122].

Los parámetros del algoritmo maximizan la detección de cavidades en estructuras de proteínas que se encuentran experimentalmente unidas a ligandos, procurando contar con la menor cantidad posible de falsos positivos. Luego de ubicar todas las esferas posibles del rango de radios determinados como parámetro del algoritmo, aquellas zonas que poseen una alta densidad de esferas superpuestas son las que poseen mayores probabilidades de alojar ligandos. Cada bolsillo se delimita entonces a través las esferas que lo conforman. A su vez, se define un parámetro en el algoritmo para determinar la mínima cantidad de esferas que son necesarias para constituir un pocket, evitando de esta manera definir cavidades muy pequeñas. De esta manera, las propiedades de cada uno de los bolsillos, vienen dadas por las características de las esferas que lo componen (volumen, superficie, etc.) y por los átomos de la proteína que contribuyeron a definir sus esferas (carga, dadores y aceptores de puente de hidrógeno, etc.). Maximizados los parámetros de corrida del algoritmo para la detección de verdaderas cavidades que alojan ligandos, se define el Druggability Score (puntaje de drogabilidad estructural) como una combinación lineal de las propiedades calculadas normalizadas entre cero y uno, de forma de representar valores más altos para bolsillos con mayores chances de alojar ligandos y valores menores cuando las chances de acoplamiento bajan.

Con respecto a las ventajas que presenta el algoritmo fpocket, podemos remarcar la variada cantidad de descriptores que el programa detalla para cada una de las cavidades. Dado que el puntaje final de drogabilidad es una función lineal de estos descriptores derivada sobre un conjunto extensivo de estructuras unidas a moléculas tipo droga — extrayendo valores óptimos para cavidades en las que se observa la unión en contraste con los descriptores de otras cavidades presentes que se conoce que no conforman un sitio de unión — es que podemos describir a este puntaje presentado como confiable. Su uso se ha hecho extensivo en la comunidad [c] tanto para estudios detallados de proteínas en particular, como en múltiples estrategias de identificación de blancos drogables de manera extensiva.

[c] Actualmente el trabajo cuenta con más 500 citas en GoogleScholar

Consideraciones:

Uno de los problemas que se ha señalado (Sección "Known Bugs" de la documentación de fpocket 2.0) es la no independencia de rotación/traslación del algoritmo, i.e., una misma estructura que se encuentra trasladada o rotada en el espacio, debido a la ubicación aleatoria establecida para el comienzo del escaneo presentará pockets ligeramente distintos. Si bien esto representa un problema de reproducibilidad, no tiene mayores incidencias en la determinación de los sitios drogables de una estructura. Prueba de eso es la amplia utilización de una re-implementación del algoritmo, MDPocket[123], el cual analiza la evolución de la drogabilidad de las cavidades y de cada uno de sus descriptores a lo largo de una dinámica molecular. Si los valores de los descriptores presentaran graves discontinuidades al verse ligeramente transformada o rotada una estructura (cosa que de hecho sucede a lo largo de una dinámica) serían inutilizables, sin embargo el algoritmo se ha establecido como estándar en el campo.

Propiedades de los pockets

A continuación se listan las propiedades calculadas por fpocket, con el nombre con el cual aparecen en el archivo de salida generado por la aplicación. El nombre de algunas propiedades puede diferir entre el archivo de salida y el que se encuentra en el header de los archivos PQR de cada pocket (por ejemplo, `Number of Alpha Spheres` aparece como `Number of V. Vertices`, "número de vértices de Voronoi").

- `Score:` Obtenido mediante una función que incorpora cinco propiedades: number of alpha spheres, mean local hydrophobic density, apolar alpha sphere proportion, polarity score y alpha sphere density. El objetivo de este puntaje es identificar, entre todos los pockets encontrados, a aquel con mayor probabilidad de ser el "sitio activo" (el de valor más alto).

- `Druggability Score:` Valor entre 0 y 1 asociado a cada pocket. Representa la probabilidad de unir una molécula pequeña tipo droga. Un valor de $0,5$ (umbral) indica que la unión de la droga al pocket puede ser posible. Un valor de 1 indica una muy alta probabilidad. Los bolsillos poseen múltiples características. Entre ellas podemos destacar la drogabilidad estructural, el factor o puntaje que determina la capacidad de los bolsillos de formar enlaces con compuestos tipo droga. Este valor ha sido optimizado en base a las propiedades y a los ejemplos experimentales de unión droga proteína, de manera de expresar de manera óptima la capacidad de una cavidad de alojar una molécula[124]. Es un valor que va de 0 (no-drogable) a 1 (altamente drogable).
  Este puntaje es de utilidad para una de las aplicaciones fundamentales y directas del procedimiento de encontrar bolsillos estructuralmente drogables: determinar el sitio activo de una proteína, el lugar donde presenta una actividad enzimática (que catalizan una reacción química dentro de la célula cumpliendo una función específica). Localizar el sitio activo y poder determinar sus propiedades estructurales resulta fundamental a la hora de diseñar compuestos que tengan buenas posibilidades de acoplarse a la proteína, modulando o inhibiendo la actividad de la misma dependiendo del efecto que quiera lograrse.

- α`-esfera:` esfera que contacta con cuatro átomos en su frontera y no tiene átomos internos. Los cuatro átomos están a igual distancia del centro, definiendo el radio.
  En el interior de una proteína se suelen ubicar esferas muy pequeñas; en la superficie,

esferas grandes. Las cavidades corresponden a esferas de tamaño intermedio. Para la detección de pockets, se efectúa una clusterización de esferas y se filtran por cierto rango de tamaños y propiedades de los átomos involucrados.
En el caso de fpocket, la clusterizacion se efectúa en tres pasos: En el primero se agrupan esferas mediante un criterio de distancia. Luego se agregan los clusters que tienen centros de masa muy cercanos, conformando un solo cluster más grande. Finalmente, las esferas de cada cluster se comparan con las de los otros. Si cierto número de esferas de un cluster están cercanas a cierto número de esferas de otro cluster, ambos clusters se unen en uno mayor.

- `Number of Alpha spheres:` Indica el número de α-esferas que forman el pocket. A más esferas, mayor tamaño de pocket. También se la puede encontrar como Número de Vértices de Voronoi
- `Volume`: volumen del pocket, aproximado por el espacio ocupado por las α-esferas. Se estima mediante algoritmo de MonteCarlo, en forma iterativa.
- `Mean alpha sphere radius`: radio promedio de las α-esferas usadas en el pocket.
- `Mean alp. sph. solvent access`: promedio de qué tan "enterrada" está cada α-esfera. Se calcula como la distancia desde el baricentro de la esfera (definida usando los cuatro átomos contactados) al centro de la esfera[125].
- `Flexibility`: es el promedio de los `B-Factor` de todos los átomos del pocket, normalizado. La documentación de fpocket aclara que no necesariamente refleja la flexibilidad en estructuras cristalizadas.
- `Hydrophobicity score`: promedio del score de hidrofobicidad[126] de todos los residuos del pocket.
- `Polarity score`: promedio del score de polaridad[127] de todos los residuos en el pocket.
- `Charge score`: promedio de carga de todos los aminoácidos[127] en contacto con al menos una α-esfera.
- `Volume score`: es el promedio de los scores de volumen[127] de todos los aminoácidos en contacto con al menos una α-esfera del pocket.
- `Mean local hydrophobic density Score`: medida de la presencia de zonas no polares locales. Para cada α-esfera se cuenta sus vecinas (es decir, aquellas con las que existe solapamiento) no polares. Luego se suman todas éstas y se divide por el total de esferas no polares del pocket. El resultado final es normalizado comparando con los pockets de la proteína.
- `Alpha sphere density`: medida de la densidad del pocket y qué tan "enterrado" (*buriedness*) se encuentra este. Es el promedio de las distancias entre cada par de α-esfera. Un valor pequeño indica un pocket bastante compacto y enterrado, mientras que un valor más grande indica un pocket más extendido y expuesto.
- `Maximum distance between center of mass and alpha sphere`: distancia máxima entre una α-esfera y el centro de masas del pocket

- `Apolar alpha sphere proportion`: indica la proporción de α-esferas no polares del pocket, definidas como aquellas en contacto con al menos 3 átomos no polares [128]
- `Proportion of polar atoms`: proporción de átomos polares del pocket. Fpocket define un átomo polar como aquel cuya electronegatividad es mayor a $2,7$ [129].
- `Apolar SASA`: provee una estimación del área de la superficie no polar del pocket accesible por solventes (*en inglés, solvent-accessible surface area*, basándose en información de los atomos receptores.
- `Polar SASA`: similar al anterior, pero teniendo en cuenta sólo átomos polares.
- `Total SASA`: suma de las áreas de la superficie polar y no polar, es decir, el área total del lado receptor del pocket.

A.2.8 K-means

K-means es un método de agrupamiento o clustering, que tiene como objetivo la partición de un conjunto de n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano.
Se considera K-means un algoritmo no supervisado, en donde los datos no están clasificados de antemano, es decir, no se posee un set de entrenamiento en el cual se conocen *a priori* los resultados que se esperan para cada elemento. El principal objetivo de estos métodos es encontrar estructuras o patrones ocultos en los datos.
El algoritmo asigna a cada elemento uno de los k grupos en función de la similitud de sus propiedades. El proceso es iterativo, es decir, que se va ajustando la posición del centro del cluster (también llamado centroide) en cada iteración del proceso. De esta forma los grupos se van definiendo de manera orgánica.

A.2.9 Coeficientes de correlación

En el trabajo se hace referencia a los métodos de Pearson y de Spearman. Cabe hacer una aclaración sobre éstos, y tiene que ver con que cualquiera sea el método utilizado, es recomendable complementarlo con un gráfico de dispersión, dado que la correlación puede fallar en describir adecuadamente relaciones no lineales y no monotónicas (ej: cuadrática, sinusoidal) [130].
En la Figura A.7 comparamos el valor de cada coeficiente con distintos conjuntos de datos.

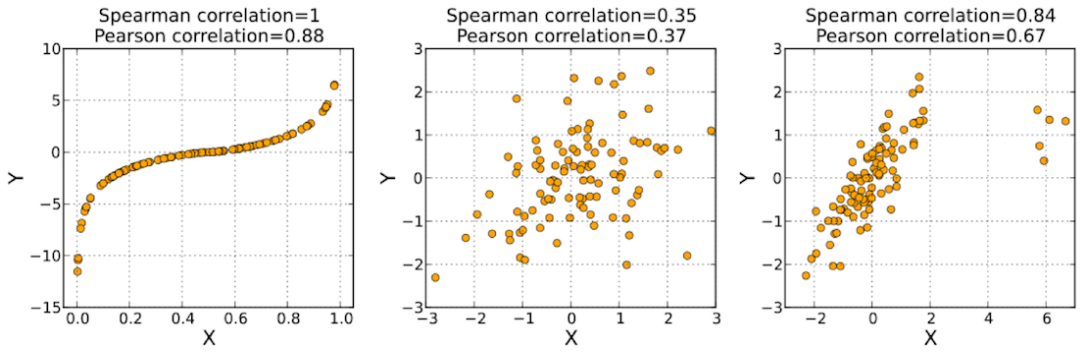


Figure A.7: Comparación de coeficientes de correlación de Spearman y Pearson entre dos variables, X e Y, calculados sobre los mismos conjuntos de datos. *Fuente: Skbkekas (CC BY-SA 3.0 Wikimedia Commons)*

A.2.10 PCA

PCA considera que la componente más interesante es la que tiene la mayor varianza o dispersión. Este razonamiento tiene una fundamento teórico: de todas las dimensiones por donde se esparcen los datos, la de mayor varianza se corresponde con la de mayor entropía. Y dado que una mayor entropía indica que se está codificando más información[131], dicha variable tiene un rol más protagónico a la hora de explicar los datos. Consecuentemente los autovectores más pequeños usualmente representarán las componentes de ruido, mientras que los de autovectores más grandes usualmente se corresponden con las componentes que condicionan fuertemente a los datos.
De modo ideal, se buscan m componentes que sean combinaciones lineales de las p propiedades originales en donde $m < p$, estando éstas no-correlacionadas. Si las variables estuviesen correlacionadas, entonces se altera todo el proceso de búsqueda, dado que refuerzan su contribución a la varianza explicada.

A.2.11 Propiedades de forma molecular

Los esferoides son formas geométricas que se obtienen al girar un elipsoide sobre uno de sus tres ejes. Pueden catalogarse en 4 categorías, como puede verse en la Figure A.8.

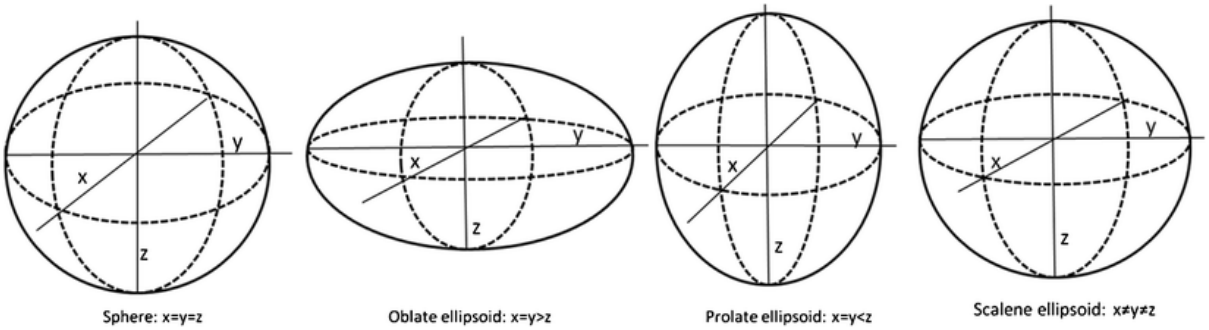


Figure A.8: Tipos de elipsoide. De izquierda a derecha son esfera, esferoide oblato, esferoide prolato y elipsoide tri-axial o escaleno

Aplicado a nuestro problema en particular, tanto la forma de los pockets como ligandos siguen

una cierta distribución, como puede detallarse en las figuras A.9 y A.10.

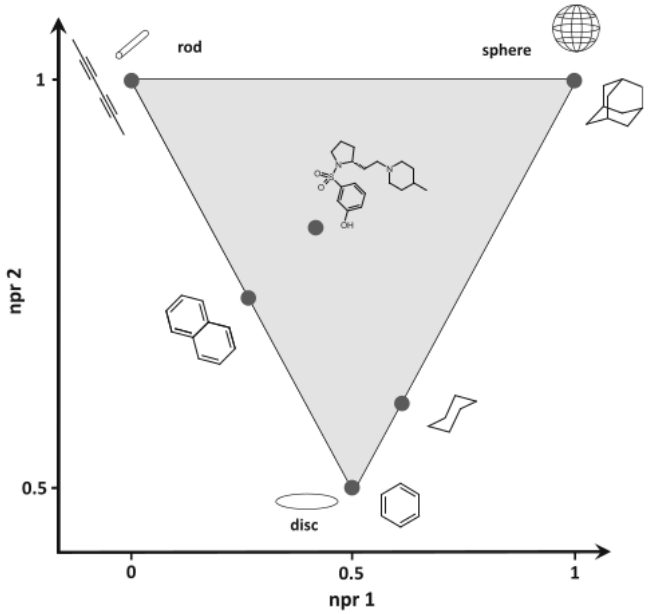


Figure A.9: Diagrama ternario de forma molecular. Las moléculas se distribuyen según su forma en este esquema, pareciéndose más o menos a los arquetipos (esfera, esferoide prolato o "palo" y esferoide oblato o "disco") mientras más cerca de los vértices se encuentren. Todos los puntos intermedios corresponden a elipsoides escalenos. *Fuente: Wirth et al.*[57]

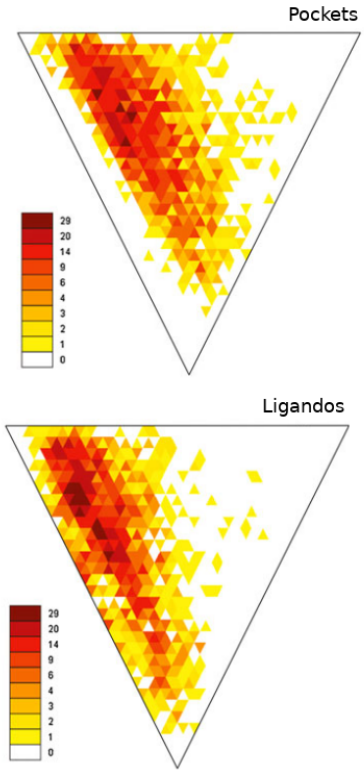
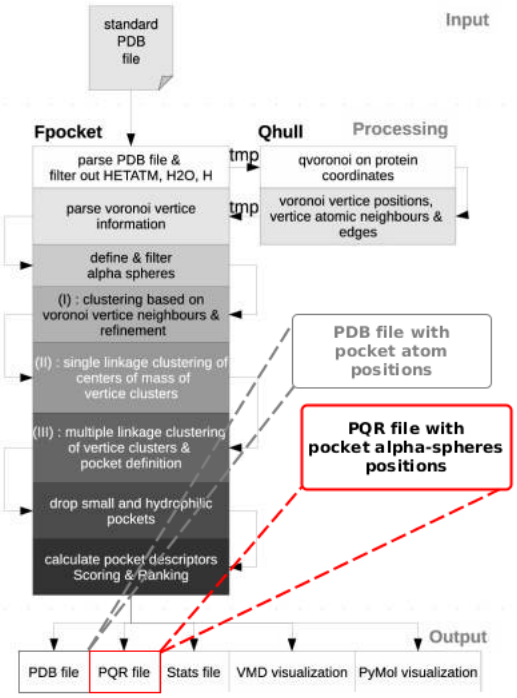


Figure A.10: Distribución de formas de pockets (superior) y ligandos (inferior) de sc-PDB[99]. La escala de colores indica la cantidad máxima de pockets (o ligandos) con esa forma, por celda. *Fuente: Wirth et al.*[57]

A.2.11.1 Pockets

El BIA nos proporcionó un conjunto de mas de 200gb de información con las corridas de fpocket que había realizado para el desarrollo de nuestro trabajo. Dentro de estos directorios encontramos cientos de miles de archivos de tipo *.pml*, *.ent*, *.ent.gz*, *.tcl*, *_VMD.sh*, *_PYMOL.sh*, *.txt*, *.out_pdb*, *_pockets.pqr* y millones de archivos *_atm.pdb* y *_vert.pqr*. Éstos últimos representan la estructura de los pockets de 2 maneras distinas.



Como se ve en la figura A.11, la corrida de fpocket genera, para cada pocket, un conjunto de archivos con información de propiedades físico-químicas y estructurales. Los archivos de tipo *_atm.pdb* (e.g. pdb200d_out/pockets/pocket0_atm.pd) definen la ubicación y tipo de átomo de todos los que componen el pocket. Mientras que los archivos *_vert.pqr* definen la ubicación y radio de las α-spheres existentes en el diagrama de Voronoi construido durante la corrida de fpocket. El archivo PQR deviene del formato PDB (sus siglas significan en inglés, **P** for pdb, **Q** for charge, **R** for radius).

Figure A.11: Diagrama de flujo del programa fpocket. *Fuente: paper fpocket* [40]

Tomamos los datos de corridas de fpocket proporcionadas por el BIA e hicimos una limpieza exhaustiva de archivos y directorios no relacionados a nuestro problema. Optamos por quedarnos únicamente con los archivos de tipo `pocket{n}_vert.pqr`, donde n es el número de pocket de la proteína. Estos archivos describen la disposición y propiedades de las alpha-spheres utilizadas durante la corrida de fpocket y se encuentran alojados en directorios que llevan en su nombre el `pdbId` de la proteína en cuestión, de forma que no hay ambigüedades a la hora de identificar a quién corresponde qué cosa.

Luego de la limpieza, quedaron únicamente 103.886 archivos. Dichos archivos fueron creados mediante la sincronización inicial de PDB, hecha mediante el script `rsyncPDB.sh`, el cual es provisto por pdb[a]. Dicho script define la estrucura que puede verse a la derecha en la figura A.12 Desarrollamos un script que recorre el árbol de directorios, encuentra los archivos `pocket{n}_vert.pqr` y los parsea.

[a]Puede encontrarse bajo la siguiente URL `ftp://ftp.wwpdb.org/pub/pdb/software/rsyncPDB.sh`.

```
.../divided/

00/ 17/ 2g/ 44/ 60/ 76/ 91/ a7/ ap/ b7/ bp/
c7/ cp/ d7/ e7/ ep/ f7/ fp/ g7/ gp/ h7/ hp/
i7/ ip/ j7/ jp/ k7/ kp/ l7/ lp/ m7/ mp/ n7/
np/ o7/ op/ p7/ pp/ q7/ qp/ r7/ rp/ s7/ sp/
t7/ tp/ u7/ up/ v7/ vp/ w7/ wp/ x7/ xp/ y7/
yp/ 01/ 18/ 30/ 45/ 61/ 77/ 92/ a8/ aq/ b8/
bq/ c8/ cq/ d8/ dq/ e8/ eq/ f8/ fq/ g8/ gq/
h8/ hq/ i8/ iq/ j8/ jq/ k8/ l8/ lq/ m8/ mq/
n8/ nq/ o8/ oq/ p8/ pq/ q8/ qq/ r8/ rq/ s8/
sq/ t8/ tq/ u8/ uq/ v8/ vq/ w8/ wq/ x8/ xq/
```

Figure A.12: Árbol de directorios con las corridas de `fpocket`.

A.2.11.2 Problemas durante la generación de MVEE para pockets

Durante la corrida del algoritmo para el dataset incial, para todas las cavidades, nos encontramos con varios casos en los cuales no se pudo realizar el cálculo debido a que los datos eran inconsistentes. Las matrices que representaban los puntos en el espacio de las α-spheres eran singulares (979 casos) o presentaban errores del tipo `LinAlgError('SVD did not converge')` (1 caso).

A.2.11.3 Ligandos

En la web de descargas de RCSB puede encontrarse el archivo SDF con la información estructural de todas las moléculas existentes en la base de datos, de forma que lo tomamos como input para nuestros scripts. Puede obtenerse o bien, el archivo que describe la estructura "ideal", o bien, el que describe la estructura surgida de un modelo. Las coordenadas de la estructura "ideal" son calculadas por software (de modo similar a como hicimos con los ligandos de ChEMBL), mientras que las coordenadas de la estructura "model" son las obtenidas experimentalmente[132] (por ejemplo, compuestos co-cristalizados con una proteína). En nuestro caso decidimos quedarnos con la estructura surgida de un modelo del compuesto, porque es probablemente la conformación que adopte el ligando en el acoplamiento[132].

Información volumétrica de ligandos

La generación de la estructura 3D en base a la estructura 2D del archivo SDF se realiza con la función `make3D`, cuya implementación en python es ofrecida por el software OpenBabel[133] utilizando MMFF94[134,135,136,137,138] como *force field* por defecto. Se pueden presentar algunos inconvenientes con los *rotable bonds* detallado en la propiedad NRB, *number of rotable bonds*), dado que puede ser complicado decidirse por una estructura molecular determinada si la misma puede presentar múltiples conformaciones. El método utilizado proporciona una precisión aceptable para compuestos tipo droga[139], sin embargo puede ser optimizado posteriormente para alcanzar una conformación de mínima energía (idealmente un mínimo global).

A.2.12 Extensión del dataset usando ChEMBL

Para trabajar con la versión de ChEMBL más actualizada, descargamos la base de datos `sql` en la página web oficial de ChEMBL. Levantamos la base de datos en una instalación local de PostgreSQL y cruzamos las tablas `public.compound_properties` con `public.molecule_dictionary` para conseguir las propiedades medibles de ligandos con su correspondiente `chembl_id`. Al igual que todos los dataset anteriores, esta información fue guardada en un archivo `csv`.
El circuito completo puede visualizarse en la figura A.13.

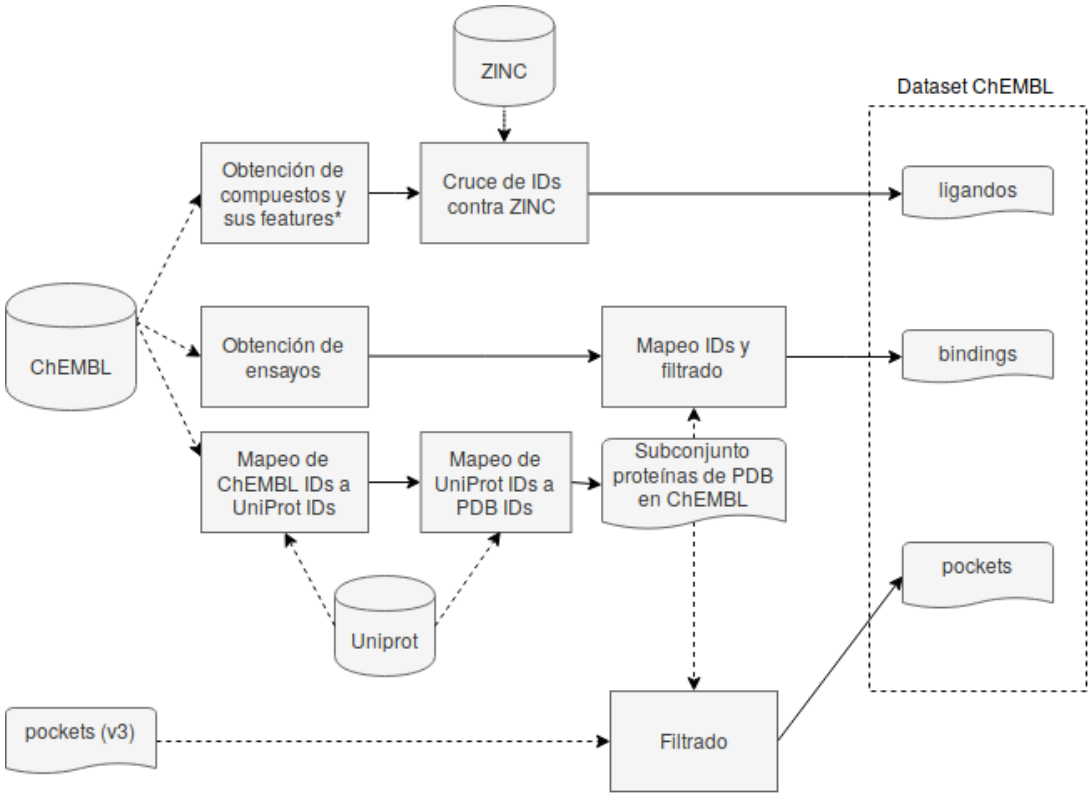


Figure A.13: Generación del dataset con base en ChEMBL. *(*) Exceptuando* `charge` *porque no se encuentra en la BD de ChEMBL*

A.2.13 Generación de features volumétricas y extensión de datasets

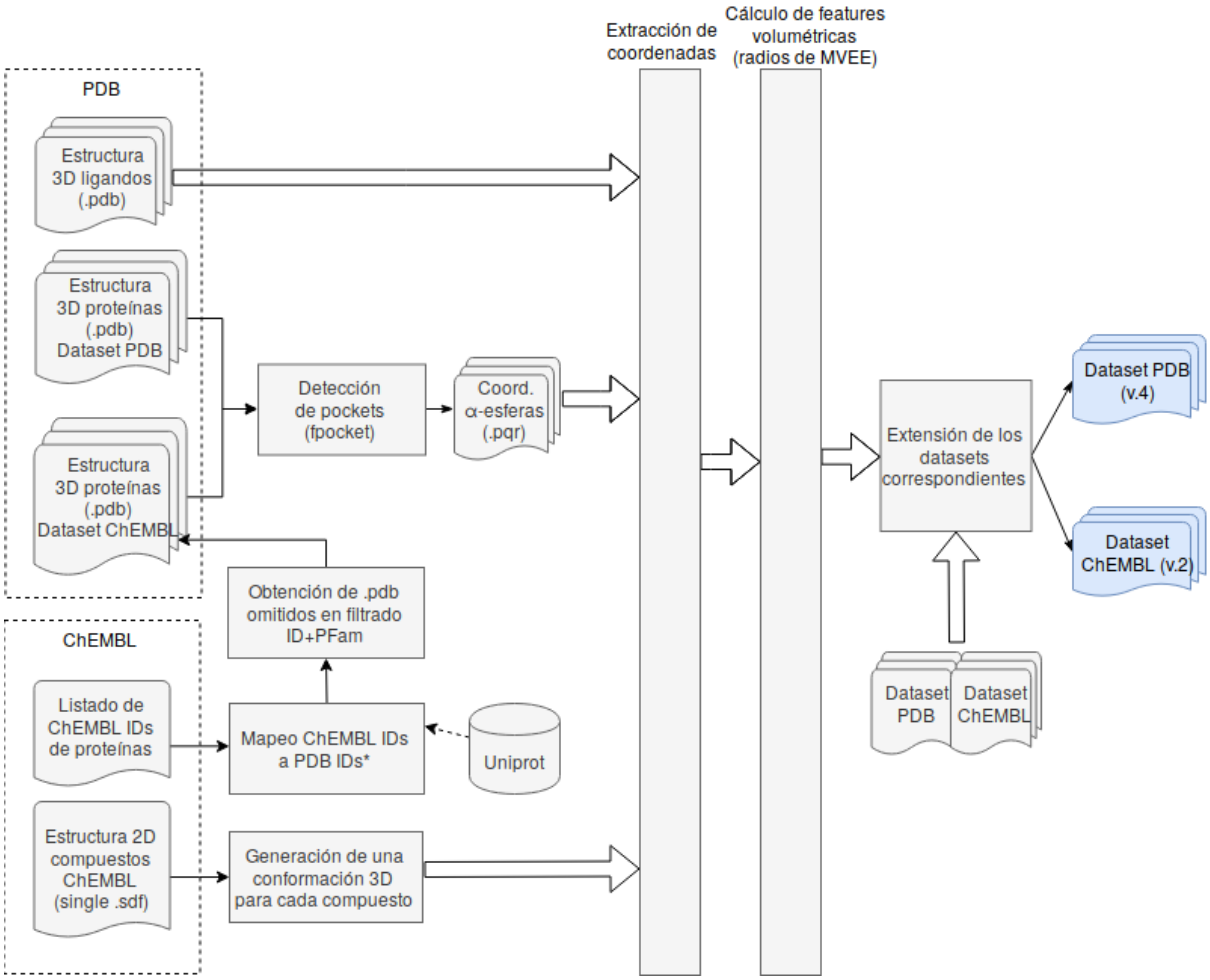


Figure A.14: Generación de features volumétricas y enriquecimiento de los datasets. *(*) Es el mismo proceso que en el diagrama anterior.*

A.2.14 Informacion residual

Los datos de acoplamientos del dataset inicial (presentes en el archivo `bindings.csv`) tienen la estructura que se describe en la tabla A.3.

protein_id	protein_model	protein_chain	protein_pocket_id
3KD1	0	A	21

Table A.3: Tabla con datos de prockets del dataset incial.

Tanto `protein_model` como `protein_chain` fueron introducidas durante el procesamiento de datos hecho por el BIA. Se utilizaron múltiples combinaciones de modelos y cadenas para calcular los posibles acoplamientos. Podría decirse que son información residual de ese procesamiento y que no otorgan un real valor para nuestros algoritmos de recomendación. Tampoco reviste utilidad para el pipeline general de LigQ, dado que éste usa las features tal cual vienen del output

de fpocket. Debido a esto optamos por quedarnos como dato útil final a los datos descriptos en la tabla A.4, entre los cuales se suma el `unified_pocket_id` descripto en la sección 2.2.1 del Capítulo 2 (Dataset).

protein_id	protein_pocket_id	unified_pocket_id
3KD1	21	8249280418640935242

Table A.4: Tabla con datos de prockets del dataset incial.

Para esto debimos tomar los recaudos suficientes y validar que no existiesen registros duplicados, que se diferencien únicamente por estos valores. Este concepto puede visualizarse en los esquema de la figura A.15.

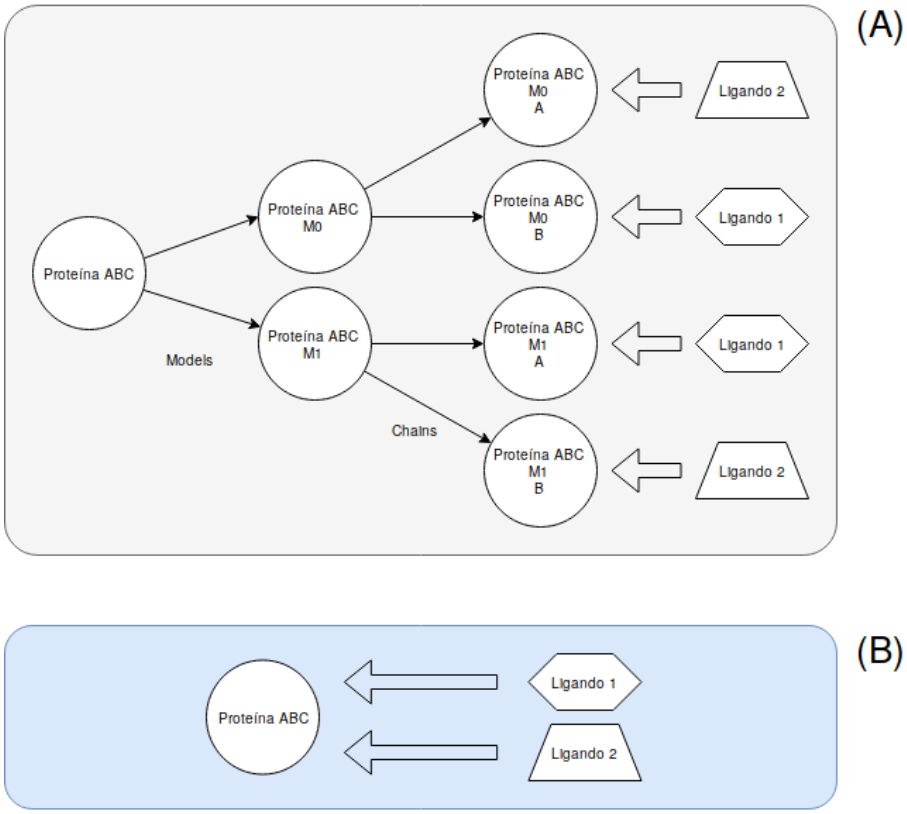


Figure A.15: En A se visualizan los bindings correspondientes a cada modelo y cadena de una misma proteína (dataset sin curar). En B Se asume que los bindings corresponden a la misma proteína.

A.3 Predicción de proteína-ligando

A.3.1 PU Learning

Positive-Undefined Learning o PU Learning[140] es una técnica de clasificación en la que dado un conjunto de ejemplos de una clase P (llamada clase *positiva*) y un conjunto de ejemplos U

no clasificados, el cual contiene tanto instancias de la clase P como la clase "no-P", el objetivo es armar un clasificador binario para clasificar el conjunto de prueba T en dos clases - positivos y negativos - donde T puede estar enteramente conformado por casos no clasificados.

A.3.2 Implementación de MinMaxScaler

Este estimador escala y transforma cada feature individualmente, provocando que se ubique dentro del rango del conjunto de entrenamiento (por ejemplo entre 0 y 1). La transformación se encuentra dada por la ecuación descripta en ecuación A.1, en donde X es la entrada y min/max definen el rango.

$$\begin{aligned} &1.X_{std} = (X - X_{min})/(X_{max} - X_{min}) \\ &2.X_{scaled} = X_{std} * (max - min) + min \qquad \text{donde } scale = (max - min)/(X_{max} - X_{min}) \\ &3.X_{scaled} = scale * X + min - X_{min} * scale \end{aligned} \tag{A.1}$$

A.3.3 Métricas de problemas de regresión

A continuación se listan las métricas más comúnmente utilizadas en problemas de regresión:

- **ERROR MEDIO ABSOLUTO (EMA):** Esta métrica de regresión es el valor medio de la diferencia absoluta entre el valor real y el valor predicho.
- **ERROR CUADRÁTICO MEDIO (ECM):** El error cuadrático medio (ECM) calcula el valor medio de la diferencia al cuadrado entre el valor real y el predicho para todos los puntos de datos. Todos los valores relacionados se elevan a la segunda potencia, por lo tanto, todos los valores negativos no se compensan con los positivos. Además, debido a las características de esta métrica, el impacto de los errores es mayor. Por ejemplo, si el error en nuestros cálculos iniciales es de 1/2/3, el ECM será igual a 1/4/9 respectivamente. Cuanto menor sea el ECM, más precisas serán nuestras predicciones. ECM = 1 es el punto óptimo en el que nuestro pronóstico es perfectamente preciso.
  El ECM tiene algunas ventajas frente al EMA: en primer lugar destaca grandes errores entre los pequeños. Por otro lado, es diferenciable, lo que ayuda a encontrar los valores mínimos y máximos utilizando los métodos matemáticos de manera más efectiva.
- **RAÍZ DEL ERROR CUADRÁTICO MEDIO (RECM):** El RECM es la raíz cuadrada del ECM. Es fácil de interpretar en comparación con el ECM y utiliza valores absolutos más pequeños, lo que es útil para los cálculos informáticos. En comparación con la Desviación Media Absoluta o MAD, RMSE amplifica y penaliza con mayor fuerza aquellos errores de mayor magnitud. La fórmula de cálculo del RMSE se muestra en la Equation A.2, donde A_t es el valor observado y F_t son los valores predecidos, mientras que n es la cantidad de muestras[d]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_t - F_t)^2} \tag{A.2}$$

[d] En Keras, la función retorna un tensor escalar:`return K.mean(K.square(y_pred - y_true), axis=-1)`.

A.3.4 Métrica utilizada por K-means: inercia

La inercia es una medida de coherencia intra-cluster, utilizada por K-means como función de pérdida. Está definida por la siguiente fórmula:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

en donde $x_0, ..., x_n$ es el conjunto total de muestras, μ_j es el centroide alguno de los clusters C tal que la distancia a la muestra x_i es mínima.

A.3.5 Features de ajuste por Grid Search

En el siguiente listado detallamos algunas particularidades de los hiperpetámetros más frecuentemente utilizados en el ajuste de una red neuronal:

`batch_size`: El tamaño del lote en el método descenso de gradiente iterativo es el número de patrones que se muestran a la red antes de actualizar los pesos. También es una optimización en el entrenamiento de la red, definiendo cuántos patrones leer a la vez y guardar en memoria.

`epochs`: El número de épocas es el número de veces que el conjunto de datos de entrenamiento se muestra a la red durante el entrenamiento. Algunas redes son sensibles al tamaño del lote, como las redes neuronales recurrentes LSTM y las redes neuronales convolucionales.

`init_mode`: Si bien la inicialización aleatoria de los pesos puede ser un valor por defecto más que adecuado para muchos casos, dado que las funciones de activación pueden variar entre capa y capa de la red neuronal, puede ser mejor usar diferentes esquemas de inicialización.

`dropout_rate`: El concepto de Dropout apareció recientemente en un trabajo titulado *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* [89]. La idea que plantea ignorar adrede nodos de la red (tanto de las capas ocultas como las visibles), así estaremos mitigando riesgos de overfitting.

`weight_constraint`: parámetro de la función `MaxNorm()`, la cual mantiene acotada la norma de los pesos que inciden en cada neurona.

`optimizer`: El optimizador ADAM es un algoritmo especializado de descenso de gradiente que utiliza tanto el gradiente calculado, como sus estadísticas y valores históricos, para dar pequeños pasos en dirección opuesta dentro del espacio del parámetro de entrada de manera de minimizar una función. Se utiliza para la optimización en el entrenamiento de redes neuronales. En otras palabras, el optimizador ADAM necesita usar un algoritmo como el algoritmo de retropropagación para primero calcular el gradiente de la función. Entonces el optimizador ADAM usar el restulado de este cálculo para realizar, en una segunda etapa, el descenso de gradiente de manera especializada.

A.3.6 BallTree

Un árbol de bolas (más conocido como *Ball Tree*) es un tipo específico de estructura de datos geométricos. En ella se organizan los datos en formas "esféricas", que ayudan con su análisis y posible usos. En la figura A.16 se muestra una visualización de estas estructuras.

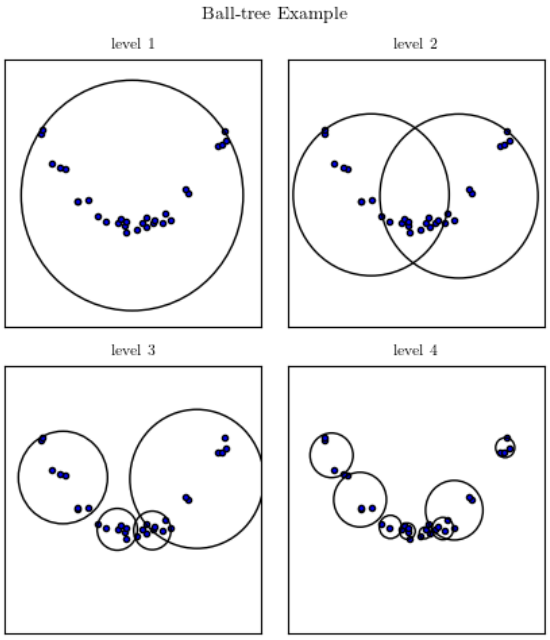


Figure A.16: Estructura de **bolas** y **nodos** del BallTree. En cada imagen se ve como las bolas envuelven cada vez menos cantidad de muestras bajo cierta lógica de conjuntos cercanos.

En la figura A.16 se muestra un conjunto de nodos circulares anidados uno dentro del otro, con las bolas más pequeñas anidadas en cada nodo. La estructura de datos en sí es un árbol que se encuentra conformado por una serie de **bolas** y **nodos**, donde el nodo interno, un nodo dentro de un nodo, se distingue por el área que incluye todas sus bolas derivadas. En particular, si el algoritmo está buscando la estructura de datos con un punto X, y ya ha visto algún Y que está más cerca de X entre los puntos encontrados hasta ahora, entonces cualquier subárbol cuya bola esté más lejos de X que Y puede ignorarse para el resto de la búsqueda. En este sentido puede ser utilizado como una forma de optimizar el cálculo de distancias entre puntos en el espacio, dado que permite ahorrar el cálculo de distancias de muchos otros puntos pertenecientes al conjunto de muestras totales.

A.3.7 Esquema de validación alternativo

Basados en trabajos anteriores[47], consideramos el siguiente esquema de actividades evaluar la calidad predictiva de nuestros modelos:

1. Remover del dataset **total** (sin considerar particiones de train/test) un pocket y los acoplamientos asociados a éste. No borramos los ligandos asociados porque aún podrían tener interacciones con otros pockets, y en caso de no tenerlas, serían descartados en el siguiente paso.
2. Entrenar el modelo de regresión.
3. Clusterizar el conjunto de ligandos.
4. Predicción del ligando ideal del pocket removido usando el regresor.
5. Determinar a qué cluster pertenece el ligando ideal.
6. Seleccionar ligandos similares. En esta etapa de evaluación se evalúa la calidad de las recomendaciones variando el valor de N, desde 1 hasta el tamaño del cluster (ver siguiente paso).
7. Evaluación de la calidad de las recomendaciones. La métrica de performance podría ser la curva ROC, definiendo como verdaderos positivos aquellos ligandos de la recomendación

que están dentro de los ligandos que se acoplan originalmente al pocket de prueba, y falsos positivos a aquellos que no.

Lamentablemente, en nuestro caso, este esquema sufre de algunos inconvenientes para los datasets PDB y 8L: debido a la distribución de bindings, no encontramos pockets con una cantidad suficiente de ligandos acoplados. Respecto a este punto, trabajamos con la siguiente hipótesis del dominio del problema: estructuras parecidas pueden dar lugar a pockets parecidos, de forma que, el hecho de haber reducido el sesgo experimental durante el curado del dataset, posiblemente haya quitado pockets similares, y por lo tanto presentan pocos acoplamientos.
Por otro lado, en el trabajo citado anteriormentes[47] se planteó la idea de que el sistema de recomendación, en su forma final, tiene que ayudar al usuario final a encontrar ligandos con buenas probabilidades de acoplarse a una proteína (no importa en qué pocket). En este sentido, en ese trabajo, el hecho de que pocos pockets tengan una cantidad no muy abundante de acoplamientos con ligandos, fue resuelto extendiendo la validación a las proteínas y todos los pockets que en ella figuran. Se basó además en proteínas con una buena cantidad de ligandos acoplados, para poder contar con más elementos que permitiesen una correcta validación del modelo. El conjunto de 4 proteínas elegidas (a través de sus bolsillo) se acoplaban a aproximadamente 250 ligandos cada una. Lamentablemente, en nuestro dataset tampoco contamos con proteínas que tengan un número tan elevado de ligandos acoplados. En este sentido, si bien dicho esquema no es aplicable tal cual a nuestro caso de estudio planteamos una alternativa manteniendo el concepto de clustering de pockets y el análisis de pocketoids, planteado en la sección 2.2.2.4 del Capítulo 2. Debido a que trabajamos con la hipótesis de que los pockets suelen parecerse mucho entre ellos, optamos por buscar coincidencias de ligandos recomendados con ligandos del dataset que tengan bindings con pockets *similares* al pocket target. Si bien parece factible la utilización de esta técnica, establecer el valor de k para el clustering de pockets que pueda tener sentido en el dominio del problema abre una nueva problemática que exige un conocimiento más profundo del área, de la herramienta `fpocket` y de las bases de datos públicas consultadas.
Como contrapartida, el hech de que estos esquemas se basen hacer mediciones utilizando "el mejor caso" puede ser un tanto engañoso como métrica real del rendimiento del recomendador. En cierta forma se focaliza en sobconjunto específico de muestras del dataset muy buena en términos cuantitativos y, si bien es cierto que puede esperarse a futuro un cierto crecimiento de los datos disponibles (y por ende más casos éstos), no permite dar una métrica concisa del rendimiento de los modelos elegidos en el momento de su construcción. Es por eso que optamos por la alternativa de utilizar particiones balanceadas como las descriptas en la versión definitiva.

B | Apéndice: LigQ y reverse-LigQ

B.1 Módulos internos y software de terceros

A continuación se listan las herramientas involucradas en LigQ:

- Servidor web
- LigQ Core (ver sección La Herramienta LigQ)
- Herramientas externas:
- Fpocket (`http://fpocket.sourceforge.net/`)
- Modeller (`https://salilab.org/modeller/`) Requiere de una clave
- JChem (`https://www.chemaxon.com/`) Requiere de una licencia
- Biblioteca desarrollada por la compañía textbfChemAxon. La misma es de uso gratuito para investigación y paga
- para usos comerciales. Permite calcular propiedades de ligandos.
- rDock (`http://rdock.sourceforge.net/`)
- rDock is a fast and versatile Open Source docking program that can be used to dock small molecules against proteins and nucleic acids.
- BLAST+ (`https://www.ncbi.nlm.nih.gov/books/NBK279690/`)
- OpenBabel (`http://openbabel.org/`)
- Bases de datos:
- zinc_specs (base de datos de compuestos, personalizada)
- all.fs, all.rmi
- JChemUtilities.jar
- pdb_seqres (utilizado para crear una db BLAST)

C | Apéndice: material suplementario

C.1 Notebooks ipython (Jupyter)

C.2 Histogramas

Todos los histogramas de las propiedades de pockets y ligandos pueden encontrarse en el directorio X.

C.3 Scripts

- `split_sdf.py`

C.4 Lista de archivos

- `Lista de cofactores y cosolventes.docx`: listados con los identificadores de PDB de compuestos excluidos del conjunto de ligandos. Son cosolventes y cofactores. Cuenta con un total de 869 registros.

Bibliography

[1] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is Bioinformatics? A Proposed Definition and Overview of the Field," *Methods of Information in Medicine*, vol. 40, no. 04, p. 346–358, 2001.

[2] R. Blumenthal, P. Rice, and R. Roberts, "Computer programs for nucleic acid sequence manipulation," *Nucleic Acids Research*, vol. 10, no. 1, p. 91–101, 1982.

[3] R. Doolittle, "Similar amino acid sequences: chance or common ancestry?," *Science*, vol. 214, no. 4517, p. 149–159, 1981.

[4] K. Sjolander, "Phylogenomic inference of protein molecular function: advances and challenges," *Bioinformatics*, vol. 20, no. 2, p. 170–179, 2004.

[5] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates, "Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles," *Proceedings of the National Academy of Sciences*, vol. 96, no. 8, p. 4285–4288, 1999.

[6] C. Sander and R. Schneider, "Database of homology-derived protein structures and the structural meaning of sequence alignment," *Proteins: Structure, Function, and Genetics*, vol. 9, no. 1, p. 56–68, 1991.

[7] J. M. Levin, B. Robson, and J. Garnier, "An algorithm for secondary structure determination in proteins based on sequence similarity," *FEBS Letters*, vol. 205, no. 2, p. 303–308, 1986.

[8] R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 32, no. 5, p. 1792–1797, 2004.

[9] L. J. Mcguffin, K. Bryson, and D. T. Jones, "The PSIPRED protein structure prediction server," *Bioinformatics*, vol. 16, no. 4, p. 404–405, 2000.

[10] S. J. Dixon and B. R. Stockwell, "Identifying druggable disease-modifying gene products," *Current Opinion in Chemical Biology*, vol. 13, no. 5-6, p. 549–555, 2009.

[11] P. Imming, C. Sinning, and A. Meyer, "Drugs, their targets and the nature and number of drug targets," *Nature Reviews Drug Discovery*, vol. 5, no. 10, p. 821–834, 2006.

[12] S. Badui Dergal, *Quimica de los Alimentos, 4ta edición*. Pearson Educación, México, 2006.

[13] K. Patton, *Anatomy and Physiology. «capítulo 4»*. Elsevier Health Sciences, 2015.

[14] H. Muirhead and M. F. Perutz, "Structure Of Hæmoglobin: A Three-Dimensional Fourier Synthesis of Reduced Human Haemoglobin at 5.5 Å Resolution," *Nature*, vol. 199, no. 4894, p. 633–638, 1963.

[15] D. Voet, J. Voet, and C. Pratt, *Fundamentos De Bioquimica. La vida a nivel molecular (2da Ed.)*. Editorial Médica Panamericana, 2007.

[16] Biology LibreTexts - OpenStax CNX, "Proteins." https://bio.libretexts.org/Bookshelves/Microbiology/Book%3A_Microbiology_(OpenStax)/07%3A_Microbial_Biochemistry/7.4%3A_Proteins. Último acceso 23 de Agosto de 2019.

[17] V. N. Uversky and A. K. Dunker, "Understanding protein non-folding," *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics*, vol. 1804, no. 6, p. 1231–1264, 2010.

[18] J. An, M. Totrov, and R. Abagyan, "Comprehensive Identification of "Druggable" Protein Ligand Binding Sites," *Genome Informatics*, vol. 15, no. 2, pp. 31–41, 2004.

[19] A. C. Anderson, "The Process of Structure-Based Drug Design," *Chemistry & Biology*, vol. 10, no. 9, p. 787–797, 2003.

[20] R. Sánchez and A. Šali, "Comparative Protein Structure Modeling: Introduction and Practical Examples with Modeller," *Protein Structure Prediction: Methods and Protocols*, vol. 143, p. 97–129, 2000.

[21] F. Paul and T. R. Weikl, "How to Distinguish Conformational Selection and Induced Fit Based on Chemical Relaxation Rates," *PLOS Computational Biology*, vol. 12, no. 9, 2016.

[22] M. S. Salahudeen and P. S. Nishtala, "An overview of pharmacodynamic modelling, ligand-binding approach and its application in clinical practice," *Saudi Pharmaceutical Journal*, vol. 25, no. 2, p. 165–175, 2017.

[23] J.-P. Kan, "Measurement And Expression Of Drug Effects," *The Practice of Medicinal Chemistry*, p. 41–49, 2003.

[24] A. Biela, M. Khayat, H. Tan, J. Kong, A. Heine, D. Hangauer, and G. Klebe, "Impact of Ligand and Protein Desolvation on Ligand Binding to the S1 Pocket of Thrombin," *Journal of Molecular Biology*, vol. 418, no. 5, p. 350–366, 2012.

[25] A. Ruiz-Garcia, M. Bermejo, A. Moss, and V. G. Casabo, "Pharmacokinetics in Drug Discovery," *Journal of Pharmaceutical Sciences*, vol. 97, no. 2, pp. 654 – 690, 2008.

[26] D. B. Kassel, "Applications of high-throughput ADME in drug discovery," *Current Opinion in Chemical Biology*, vol. 8, no. 3, pp. 339 – 345, 2004.

[27] T. Takenaka, "Classical vs reverse pharmacology in drug discovery," *BJU International*, vol. 88, p. 7–10, 2008.

[28] B. D. Anson, J. Ma, and J. Q. He, "Identifying cardiotoxic compounds," *Genetic Engineering and Biotechnology News*, vol. 29, pp. 34–35, 2009.

[29] W. Walters, M. T. Stahl, and M. A. Murcko, "Virtual screening—an overview," *Drug Discovery Today*, vol. 3, no. 4, pp. 160 – 178, 1998.

[30] F. Saldívar-González, F. D. Prieto-Martínez, and J. L. Medina-Franco, "Descubrimiento y desarrollo de fármacos: un enfoque computacional," *Educación Química*, vol. 28, no. 1, pp. 51 – 58, 2017.

[31] K. T. Schomburg, S. Bietz, H. Briem, A. M. Henzler, S. Urbaczek, and M. Rarey, "Facing the Challenges of Structure-Based Target Prediction by Inverse Virtual Screening," *Journal of Chemical Information and Modeling*, vol. 54, no. 6, pp. 1676–1686, 2014. PMID: 24851945.

[32] W. L. Jorgensen, "The Many Roles of Computation in Drug Discovery," *Science*, vol. 303, no. 5665, p. 1813–1818, 2004.

[33] X. Xu, M. Huang, and X. Zou, "Docking-based inverse virtual screening: methods, applications, and challenges," *Biophysics Reports*, vol. 4, no. 1, p. 1–16, 2018.

[34] J. A. Lee, M. T. Uhlik, C. M. Moxham, D. Tomandl, and D. J. Sall, "Modern Phenotypic Drug Discovery Is a Viable, Neoclassic Pharma Strategy," *Journal of Medicinal Chemistry*, vol. 55, no. 10, p. 4527–4538, 2012.

[35] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook," *Recommender Systems Handbook*, p. 1–35, 2010.

[36] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, pp. 61–70, dec 1992.

[37] L. Radusky, S. Ruiz-Carmona, C. Modenutti, X. Barril, A. G. Turjanski, and M. A. Martí, "LigQ: A Webserver to Select and Prepare Ligands for Virtual Screening," *Journal of Chemical Information and Modeling*, vol. 57, no. 8, p. 1741–1746, 2017.

[38] "Plataforma Bioinformática Argentina (BIA)." `http://http://www.biargentina.com.ar/`. Último acceso 4 de Septiembre de 2019.

[39] AltexSoft, "Preparing Your Dataset for Machine Learning: 8 Basic Techniques That Make Your Data Better." `https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-bett` 2017. Último acceso 28 de Agosto de 2019.

[40] V. Le Guilloux, P. Schmidtke, and P. Tuffery, "Fpocket: An open source platform for ligand pocket detection," *BMC Bioinformatics*, vol. 10, p. 168, Jun 2009.

[41] "HMMER: biosequence analysis using profile hidden Markov models." `http://hmmer.org/`. Último acceso 26 de Agosto de 2019.

[42] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, p. 1658–1659, 2006.

[43] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, "CD-HIT: accelerated for clustering the next-generation sequencing data," *Bioinformatics*, vol. 28, p. 3150–3152, Nov 2012.

[44] Project Jupyter. `https://jupyter.org/`. Último acceso 26 de Agosto de 2019.

[45] B. Ma, M. Shatsky, H. J. Wolfson, and R. Nussinov, "Multiple diverse ligands binding at a single protein site: A matter of pre-existing populations," *Protein Science*, vol. 11, p. 184–197, Jan 2009.

[46] Scikit-learn. `https://scikit-learn.org/`. Último acceso 26 de Agosto de 2019.

[47] F. Leto Mera, " Aprendizaje automático para la predicción de potenciales ligandos frente a nuevos blancos proteicos," Master's thesis, Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires, 2017. Directores: Dr. Martí, Marcelo - Lic. Radusky, Leandro.

[48] L. F. Restrepo B and J. A. GonzÃ¡lez L, "De Pearson a Spearman," *Revista Colombiana de Ciencias Pecuarias*, vol. 20, pp. 183 – 192, 06 2007.

[49] A. Ghasemi and S. Zahediasl, "Normality Tests for Statistical Analysis: A Guide for Non-Statisticians," *International journal of endocrinology and metabolism*, vol. 10, pp. 486–489, 12 2012.

[50] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. Último acceso 26 de Agosto de 2019.

[51] R. B. Dagostino, "An Omnibus Test of Normality for Moderate and Large Size Samples," *Biometrika*, vol. 58, no. 2, p. 341, 1971.

[52] R. Dagostino and E. S. Pearson, "Tests for departure from normality. Empirical results for the distributions of b_2 and $\sqrt{b_1}$," *Biometrika*, vol. 60, no. 3, p. 613–622, 1973.

[53] Laurae, "Large amount of observations: Statistical Test not so Statistical." `https://medium.com/data-design/large-amount-of-observations-statistical-test-not-so-statistical-3d8ed0e94be`, 2016. Último acceso 17 de Julio de 2019.

[54] H. Akoglu, "Users guide to correlation coefficients," *Turkish Journal of Emergency Medicine*, vol. 18, no. 3, p. 91–93, 2018.

[55] F. Song, Z. Guo, and D. Mei, "Feature Selection Using Principal Component Analysis," vol. 1, pp. 27–30, Nov 2010.

[56] K. Berlin, D. P. O'Leary, and D. Fushman, "Improvement and analysis of computational methods for prediction of residual dipolar couplings," *Journal of Magnetic Resonance*, vol. 201, no. 1, p. 25–33, 2009.

[57] M. Wirth, A. Volkamer, V. Zoete, F. Rippmann, O. Michielin, M. Rarey, and W. H. B. Sauer, "Protein pocket and ligand shape comparison and its application in virtual screening," *Journal of Computer-Aided Molecular Design*, vol. 27, no. 6, p. 511–524, 2013.

[58] Y. E. Ryabov, C. Geraghty, A. Varshney, and D. Fushman, "An Efficient Computational Method for Predicting Rotational Diffusion Tensors of Globular Proteins Using an Ellipsoid Representation," *Journal of the American Chemical Society*, vol. 128, no. 48, p. 15432–15444, 2006.

[59] M. J. Todd, "Minimum-Volume Ellipsoids: Theory and Algorithms," p. 11–23, 2016.

[60] Michael Imelfort, "ellipsoid." `https://github.com/minillinim/ellipsoid`. Licencia GPLv3. Último acceso 21 de Agosto de 2019.

[61] L. G. Khachiyan, "Rounding of Polytopes in the Real Number Model of Computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.

[62] P. Ertl, R. Lewis, E. Martin, and V. Polyakov, "In silico generation of novel, drug-like chemical matter using the LSTM neural network," 12 2017.

[63] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. P. Magariños, J. P. Overington, G. Papadatos, I. Smit, and A. R. Leach, "The ChEMBL database in 2017," *Nucleic Acids Research*, vol. 45, pp. D945–D954, 11 2016. https://www.ebi.ac.uk/chembl/.

[64] "JChem Base: structure searching and chemical database access and management." `https://chemaxon.com/products/instant-jchem`.

[65] Dr. S. Balakrishnan and Dr. A. Jebaraj Ratnakumar and J. Elumalai, "A Machine Learning Based Regression Techniques for E-Mail Prioritization," *Taga Journal of Graphic Technology*, vol. 14, p. 710–717, 2018.

[66] "Rating data sets from the MovieLens web site.." `https://grouplens.org/datasets/movielens/`. Último acceso 16 de Octubre de 2019.

[67] E. Alpaydin, *Introduction to Machine Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2014.

[68] W. Lan, J. Wang, M. Li, J. Liu, Y. Li, F.-X. Wu, and Y. Pan, "Predicting drug–target interaction using positive-unlabeled learning," *Neurocomputing*, vol. 206, pp. 50 – 57, 2016. SI:DMSB.

[69] "DUD-E - A Database of Useful Decoys: Enhanced." `http://dude.docking.org/`. Último acceso 21 de Agosto de 2019.

[70] M. Réau, F. Langenfeld, J.-F. Zagury, N. Lagarde, and M. Montes, "Decoys Selection in Benchmarking Datasets: Overview and Perspectives," *Frontiers in Pharmacology*, vol. 9, 2018.

[71] B. Das, N. C. Krishnan, and D. J. Cook, "Handling Class Overlap and Imbalance to Detect Prompt Situations in Smart Homes," pp. 266–273, Dec 2013.

[72] O. Güner, "History and Evolution of the Pharmacophore Concept in Computer-Aided Drug Design," *Current Topics in Medicinal Chemistry*, vol. 2, p. 1321–1332, Jan 2002.

[73] G. Schneider and K.-H. Baringhaus, *De Novo Design: From Models to Molecules*, ch. 1, pp. 1–55. John Wiley and Sons, Ltd, 2013.

[74] H. M. Vinkers, M. R. D. Jonge, F. F. D. Daeyaert, J. Heeres, L. M. H. Koymans, J. H. V. Lenthe, P. J. Lewi, H. Timmerman, K. V. Aken, P. A. J. Janssen, and et al., "Synopsis: Synthesize and optimize system in silico," *Journal of Medicinal Chemistry*, vol. 46, no. 13, p. 2765–2773, 2003.

[75] G. Schneider and U. Fechner, "Computer-based de novo design of drug-like molecules," *Nature Reviews Drug Discovery*, vol. 4, no. 8, p. 649–663, 2005.

[76] Y. Lecun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, p. 9–50, 1998.

[77] S. Ali and K. A. Smith-Miles, "Improved support vector machine generalization using normalized input space," *Lecture Notes in Computer Science AI 2006: Advances in Artificial Intelligence*, p. 362–371, 2006.

[78] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, "Dataset shift in machine learning," 01 2009.

[79] S. Amos, "When training and test sets are different: Characterizing learning transfer," *Dataset Shift in Machine Learning*, p. 2–28, Dec 2008.

[80] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, p. 1145–1159, 1997.

[81] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.

[82] J. Bergstra and Y. Bengio, "Random Search for Hyper-parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[83] M. Hasan, M. Mohibullah, and M. Hossain, "Comparison of Euclidean Distance Function and Manhattan Distance Function Using K-Mediods," *International Journal of Computer Science and Information Security*, vol. 13, pp. 61–71, 10 2015.

[84] F. Rosenblatt, "Principles Of Neurodynamics. Perceptrons And The Theory Of Brain Mechanisms," 1961.

[85] Keras. `https://keras.io/`. Último acceso 17 de Julio de 2019.

[86] Tensorflow. `https://www.tensorflow.org/`. Último acceso 13 de Noviembre de 2019.

[87] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.

[88] S. Bock, J. Goppold, and M. Weiß, "An improvement of the convergence proof of the ADAM-Optimizer," *ArXiv*, vol. abs/1804.10587, 2018.

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[90] H. Shen, "Towards a mathematical understanding of the difficulty in learning with feed-forward neural networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[91] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," vol. 811, 01 2011.

[92] "What is a Container? | App Containerization | Docker." `https://www.docker.com/resources/what-container`. Último acceso 25 de Octubre de 2019.

[93] G. Maggiora, M. Vogt, D. Stumpfe, and J. Bajorath, "Molecular Similarity in Medicinal Chemistry," *Journal of Medicinal Chemistry*, vol. 57, no. 8, pp. 3186–3204, 2014. PMID: 24151987.

[94] E. G. et al., *Patrones de diseño: elementos de software orientado a objetos reutilizable.* Addison-Wesley professional computing series, Pearson Educación, 2002.

[95] "PEP 8 – Style Guide for Python Code." `https://www.python.org/dev/peps/pep-0008/`. Último acceso 25 de Octubre de 2019.

[96] A. L. Hopkins, "Network pharmacology: the next paradigm in drug discovery," *Nature Chemical Biology*, vol. 4, no. 11, p. 682–690, 2008.

[97] L. M. Mayr and D. Bojanic, "Novel trends in high-throughput screening," *Current Opinion in Pharmacology*, vol. 9, no. 5, pp. 580 – 588, 2009. Anti-infectives/New technologies.

[98] J. Inglese, D. S. Auld, A. Jadhav, R. L. Johnson, A. Simeonov, A. Yasgar, W. Zheng, and C. P. Austin, "Quantitative high-throughput screening: A titration-based approach that efficiently identifies biological activities in large chemical libraries," *Proceedings of the National Academy of Sciences*, vol. 103, no. 31, pp. 11473–11478, 2006.

[99] J. Meslamani, D. Rognan, and E. Kellenberger, "sc-PDB: a database for identifying variations and multiplicity of 'druggable' binding sites in proteins," *Bioinformatics*, vol. 27, pp. 1324–1326, 03 2011.

[100] A. Volkamer, D. Kuhn, F. Rippmann, and M. Rarey, "DoGSiteScorer: a web server for automatic binding site prediction, analysis and druggability assessment," *Bioinformatics*, vol. 28, pp. 2074–2075, 05 2012.

[101] "ChEMBL - What is the 'Confidence Score'?." `https://chembl.gitbook.io/chembl-interface-documentation/frequently-asked-questions/chembl-data-questions#what-is-the-confidence-score`. Último acceso 16 de Octubre de 2019.

[102] A. Holzinger, C. Biemann, C. Pattichis, and D. B. Kell, "What do we need to build explainable AI systems for the medical domain?," 12 2017.

[103] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, vol. 1, pp. 1–10, 10 2017.

[104] M. Markou and S. Singh, "Novelty detection: a review—part 1: statistical approaches," *Signal Processing*, vol. 83, no. 12, pp. 2481 – 2497, 2003.

[105] M. P. Pollastri, "Overview on the Rule of Five," *Current Protocols in Pharmacology*, vol. 49, no. 1, pp. 9.12.1–9.12.8, 2010.

[106] "Guide to Understanding PDB Data: Methods for Determining Atomic Structures." `https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/methods-for-determining-structure`. Último acceso 17 de Julio de 2019.

[107] "PDB Data Distribution by Experimental Method and Molecular Type." `https://www.rcsb.org/stats/summary`. Último acceso 17 de Julio de 2019.

[108] E. P. Carpenter, K. Beis, A. D. Cameron, and S. Iwata, "Overcoming the challenges of membrane protein crystallography," *Current Opinion in Structural Biology*, vol. 18, no. 5, p. 581–586, 2008.

[109] S. White, "Membrane proteins of known 3D structure." `https://blanco.biomol.uci.edu/mpstruc/`. Último acceso 17 de Julio de 2019.

[110] "Modeller - Program for Comparative Protein Structure Modelling by Satisfaction of Spatial Restraints." `https://salilab.org/modeller/`. Último acceso 4 de Septiembre de 2019.

[111] C. Chothia and A. Lesk, "The relation between the divergence of sequence and structure in proteins.," *The EMBO Journal*, vol. 5, no. 4, pp. 823–826, 1986.

[112] B. Rost, "Twilight zone of protein sequence alignments," *Protein Engineering, Design and Selection*, vol. 12, no. 2, p. 85–94, 1999.

[113] E. Krieger, S. B. Nabuurs, and G. Vriend, "Homology modeling.," *Methods of biochemical analysis*, vol. 44, pp. 509–23, 2003.

[114] S. Ovchinnikov, H. Park, N. Varghese, P.-S. Huang, G. Pavlopoulos, D. Kim, H. Kamisetty, N. C. Kyrpides, and D. Baker, "Protein structure determination using metagenome sequence data," *Science*, vol. 355, pp. 294–298, 01 2017.

[115] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl, "The Protein Folding Problem," *Annual Review of Biophysics*, vol. 37, no. 1, p. 289–316, 2008.

[116] T. U. Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Research*, vol. 47, pp. D506–D515, 11 2018.

[117] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

[118] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, E. L. L. Sonnhammer, L. Hirsh, L. Paladin, D. Piovesan, S. C. E. Tosatto, and R. D. Finn, "The Pfam protein families database in 2019," *Nucleic Acids Research*, vol. 47, pp. D427–D432, 10 2018.

[119] T. Sterling and J. J. Irwin, "Zinc 15 – ligand discovery for everyone," *Journal of Chemical Information and Modeling*, vol. 55, no. 11, pp. 2324–2337, 2015. https://zinc.docking.org, Úlitmo acceso agosto 2019.

[120] J. Chambers, M. Davies, A. Gaulton, A. Hersey, S. Velankar, R. Petryszak, J. Hastings, L. Bellis, S. McGlinchey, and J. P. Overington, "UniChem: a unified chemical structure cross-referencing and identifier tracking system," *Journal of Cheminformatics*, vol. 5, p. 3, Jan 2013.

[121] D. F. Veber, S. R. Johnson, H.-Y. Cheng, B. R. Smith, K. W. Ward, and K. D. Kopple, "Molecular Properties That Influence the Oral Bioavailability of Drug Candidates," *Journal of Medicinal Chemistry*, vol. 45, no. 12, p. 2615–2623, 2002.

[122] T. Simões, D. Lopes, S. Dias, F. Fernandes, J. Pereira, J. Jorge, C. Bajaj, and A. Gomes, "Geometric Detection Algorithms for Cavities on Protein Surfaces in Molecular Graphics: A Survey," *Computer Graphics Forum*, vol. 36, p. 643–683, Jan 2017.

[123] P. Schmidtke, A. Bidon-Chanal, F. Javier Luque, and X. Barril, "MDpocket : Open Source Cavity Detection and Characterization on Molecular Dynamics Trajectories," *Bioinformatics (Oxford, England)*, vol. 27, pp. 3276–85, 10 2011.

[124] P. Schmidtke and X. Barril, "Understanding and Predicting Druggability. A High-Throughput Method for Detection of Drug Binding Sites," *Journal of Medicinal Chemistry*, vol. 53, p. 5858–5867, Dec 2010.

[125] "fpocket: descriptors.c Source File (alpha sphere solvent accessibility)." `http://fpocket.sourceforge.net/programmers_guide/descriptors_8c-source.html#l1164`. Último acceso 19 de Octubre de 2019.

[126] O. D. Monera, T. J. Sereda, N. E. Zhou, C. M. Kay, and R. S. Hodges, "Relationship of sidechain hydrophobicity and α-helical propensity on the stability of the single-stranded amphipathic α-helix," *Journal of Peptide Science*, vol. 1, no. 5, pp. 319–329, 1995.

[127] Departamento de Informática de la Universidad de Angers, "Tables de correspondance entre acides aminés et propriétés physico-chimiques." `http://www.info.univ-angers.fr/~gh/Idas/proprietes.htm`. Último acceso 13 de Octubre de 2019.

[128] "fpocket: voronoi.c Source File (apolar alpha spheres)." `http://fpocket.sourceforge.net/programmers_guide/voronoi_8c.html#l00277`. Último acceso 19 de Octubre de 2019.

[129] "fpocket: descriptors.c Source File (polar atoms)." `http://fpocket.sourceforge.net/programmers_guide/descriptors_8c.html#l00432`. Último acceso 19 de Octubre de 2019.

[130] P. Schober, C. Boer, and L. A. Schwarte, "Correlation Coefficients: Appropriate Use and Interpretation," *Anesthesia & Analgesia*, vol. 126, no. 5, p. 1763–1768, 2018.

[131] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[132] "Guide to Understanding PDB Data: Small Molecule Ligands > Ideal and Model Ligand Representations)." `https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/small-molecule-ligands`. Último acceso 16 de Octubre de 2019.

[133] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open Babel: An open chemical toolbox," *Journal of Cheminformatics*, vol. 3, p. 33, Oct 2011.

[134] T. A. Halgren, "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, p. 490–519, 1996.

[135] T. A. Halgren, "Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions," *Journal of Computational Chemistry*, vol. 17, no. 5-6, p. 520–552, 1996.

[136] T. A. Halgren, "Merck molecular force field. III. Molecular geometries and vibrational frequencies for MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, p. 553–586, 1996.

[137] T. A. Halgren and R. B. Nachbar, "Merck molecular force field. IV. conformational energies and geometries for MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, p. 587–615, 1996.

[138] T. A. Halgren, "Merck molecular force field. V. Extension of MMFF94 using experimental data, additional computational data, and empirical rules," *Journal of Computational Chemistry*, vol. 17, no. 5-6, p. 616–641, 1996.

[139] " MMFF94 Force Field (mmff94). Open Babel documentation ." `https://open-babel.readthedocs.io/en/latest/Forcefields/mmff94.html`. Último acceso 3 de Diciembre de 2018.

[140] X.-L. Li and B. Liu, "Learning from Positive and Unlabeled Examples with Different Data Distributions," *Machine Learning: ECML 2005 Lecture Notes in Computer Science*, p. 218–229, 2005.