Aplicación de minería de datos y modelamiento matemático en ingeniería de proteínas

Diseño e implementación de nuevas metodologías para el estudio de mutaciones

UNIVERSIDAD DE CHILE

David Medina Ortiz

Supervisor: Dr. Álvaro Olivera

Departamento de Ingeniería Química, Biotecnología y Materiales
Universidad de Chile

Este trabajo es para obtener el grado de
Dr. en Ciencias de la Ingeniería

July 2019

Tabla de contenidos

Lista de figuras

Lista de tablas

Chapter 1

Aplicaciones de la minería de datos en ingeniería de proteínas

La ingeniería de proteínas, es una de las ramas más relevantes y de mayor impacto en el campo de la biotecnología. Su objetivo principal, se basa en el diseño de mutaciones, enfocadas en adicionar características específicas o mejorar sus propiedades fisicoquímicas, ya sea para someterlas a distintos tipos de ambientes, adecuarla a interactuar con diferentes elementos, presentar una mayor estabilidad, etc. [116].

Los diseños de mutaciones se resumen en dos técnicas principales: El diseño racional [40] y la evolución dirigida [10], ambas técnicas experimentales, que cumplen con el mismo objetivo, relacionado a alterar las propiedades de la proteína para provocar una mejora con respecto a la estructura inicial.

A pesar de que ambas técnicas son utilizadas día a día, en diferentes investigaciones, éstas presentan limitantes importantes, en particular, relacionadas con el espacio de búsqueda posible a explorar, el tiempo que conlleva realizar diferentes pruebas y el costo económico y de recursos humanos que implica evaluar diferentes mutaciones [150].

Diferentes métodos computacionales han sido desarrollados, enfocados principalmente en el estudio de proteínas y el apoyo al diseño de mutaciones. Estas herramientas, se centran en el análisis de la estructura y la estabilidad termodinámica de sustituciones, adiciones o eliminaciones de residuos [161, 102, 137, 134]. Sin embargo, debido a cómo estos funcionan, en ocasiones, el costo computacional es muy elevado, ya que escala linealmente con respecto a la cantidad de pruebas a realizar.

Por otro lado, métodos basados en técnicas de minería de datos y aprendizaje de máquinas, se presentan como una alternativa potente y de costo computacional reducido, siendo capaces de generar resultados a partir de conocimiento existente, ya sea, para entrenar modelos que

permitan evaluar mutaciones desde puntos de vista de estabilidad, identificación de residuos relevantes en propiedades fisicoquímicas, evaluar propensiones a cambios, etc. [37, 70, 76].

No obstante, dado el gran volumen de datos existente en la actualidad ¿cómo es posible reconocer qué dato es relevante y cuál no?, ¿cómo puedo representar la información existente?, ¿cómo puedo complementar las diversas técnicas experimentales desarrolladas con el enfoque de la minería de datos?, etc., son interrogantes que nacen a partir del uso de técnicas computacionales y el apoyo a las metodologías experimentales.

Dado lo anterior, durante el presente capítulo, se expone el concepto de ingeniería de proteínas y cuáles son las principales técnicas experimentales involucradas en el diseño de mutaciones. Además se introduce el concepto de Minería de datos y se exponen diversas metodologías computacionales que han sido desarrolladas, enfocadas en este campo de investigación. Por último, se presentan diferentes problemas que dan el punto de partida a cada una de las propuestas metodológicas de esta tesis doctoral y que denotan la evidente necesidad de ser desarrolladas o que exponen cambios en los puntos de vista actuales asociados al desarrollo de modelos y las convierten en un aporte significativo a los estudios actuales de mutaciones y el diseño de proteínas.

Adicional a ello, se exponen una hipótesis y objetivos generales que se relacionan estrechamente con cada capítulo siguiente en este proyecto y que permiten visualizar los diferentes puntos de vista y utilidades asociadas a las metodologías a proponer durante este trabajo de tesis doctoral.

1.1 Ingeniería de proteínas

Por definición, la ingeniería de proteínas, es un campo de investigación centrado en el diseño y creación de proteínas útiles o con propiedades fisicoquímicas relevantes, con un enfoque relevante en la comprensión del plegamiento de proteínas [113].

Actualmente, la ingeniería de proteínas, cuenta con dos estrategias principales para la construcción de mutaciones. Siendo éstas la evolución dirigida y el diseño racional. Sin embargo, a pesar de su existencia, normalmente no son excluyentes, por lo que es común utilizar ambas metodologías. No obstante, un estudio completo de residuos y una evaluación detallada de las sustituciones es una limitante importante para estas dos técnicas, debido tanto a recursos económicos como humanos [113].

La evolución dirigida imita el proceso de selección natural, permitiendo direccionar la evolución hacia objetivos definidos, reflejados en cuanto a funciones o propiedades fisicoquímicas deseables [115, 10]. Una representación del proceso, se observa en la Figura 1.1.

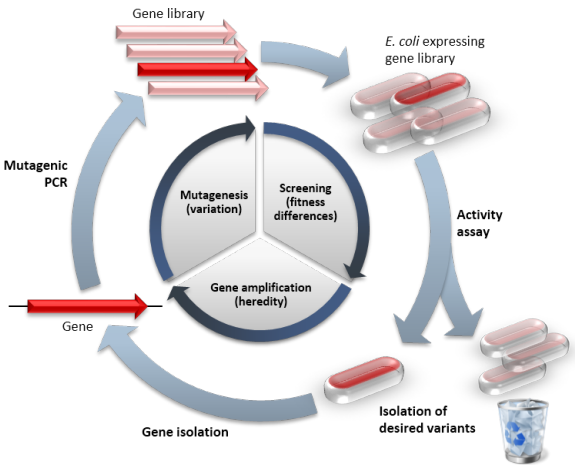


Fig. 1.1 Esquema representativo de los pasos que contempla la evolución dirigida

El proceso, de manera general, consiste en someter un gen de interés a rondas iterativas de mutagénesis, con el fin de crear una biblioteca de variantes. A partir de dicho conjunto de elementos, se seleccionan las variantes con la función deseada. Finalmente, se aíslan y se amplifican para forman una plantilla para la siguiente iteración. Así el proceso sigue iterando y estadísticamente, se seleccionan las más favorables y aquellas que tendieron a la evolución debido a la supervivencia en el proceso [10].

Con respecto al diseño racional de proteínas. Ésta, es una técnica ampliamente utilizada y al igual que la evolución dirigida, presenta el objetivo general de generar variantes con alguna función de interés o características particulares. No obstante, presenta una diferencia relevante, la cual se centra en la información que debe existir sobre la estructura, mecanismos, plegamiento o secuencia lineal de la proteína de interés [40].

1.2 Métodos computacionales aplicados en ingeniería de proteínas

Diferentes métodos computacionales han sido desarrollados para distintos análisis, con el fin de poder de poder responder diferentes interrogantes planteadas desde enfoques distintos, ya sea, para estudiar secuencias lineales, filogenia y motivos conservados, análisis de estructuras, modelamiento, estudio de mutaciones, etc. A continuación, se listan algunos de los principales enfoques y qué herramientas existen para su desarrollo.

1.2.1 Métodos de análisis filogenéticos

Los análisis filogenéticos se centran en el estudio de secuencias lineales de proteínas o genes, con el fin de identificar parentesco, motivos conservados o identificación de dominios. Las principales metodologías se basan en realizar alineamientos de secuencia con el fin de identificar o reconocer identidades de la secuencia estudio con respecto a información reportada previamente.

Una de las herramientas más conocidas para el desarrollo de alineamientos es Blast [101], el cual permite hacer múltiples comparaciones de secuencias versus bases de datos con genes o proteínas, empleando algoritmos de alineamiento local [5].

Por otro lado, se encuentran los alineamientos múltiples, los cuales son utilizados en la comparación de secuencias lineales con el fin de encontrar patrones o motivos conservados, o, la identificación de parentesco [166]. Esto último, es muy utilizado cuando se analizan secuencias de organismos no secuenciados y cuya función se trata de aclarar. Una de las herramientas más utilizadas en esta área es el software Mega [164].

A su vez, el uso de los conceptos filogenéticos, ha sido utilizado para el estudio de propensiones de sustituciones aminoacídicas y cómo éstas afectan a la función de la proteína o con respecto a sus propiedades fisicoquímicas. Considerando esto, herramientas como MOSST [134], han permitido comprender la propensión de residuos y posiciones relevantes en secuencias, sólo estudiando conjuntos de elementos sin la necesidad de conocer estructuras tridimensionales de las proteínas.

Estos estudios, generan las bases para el análisis de secuencias y se basan en que sólo se necesita la secuencia lineal a estudiar y a partir de ella, es factible comprender un panorama relacionado a patrones de conservación, tendencias, relaciones evolutivas o inclusive, propensiones y posiciones relevantes. No obstante, no son los únicos, ya que existen herramientas computacionales que facilitan la predicción de la estructura secundaria, funcionalidad, etc., siendo una de las principales herramientas Swiss-Prot [21].

1.2.2 Métodos de análisis de estructuras

Los métodos de análisis de estructuras, tienen el objetivo de comprender patrones de interacción, efectos de energía y estudiar diferentes propiedades fisicoquímicas y termodinámicas, a partir de la estructura tridimensional de una proteína, la cual puede ser obtenida por cristalografía de rayos X o por medio de resonancia magnética nuclear.

No obstante, también, es factible el desarrollo de modelos de proteínas a partir de secuencias lineales, técnica conocida como Modelamiento por homología. Diferentes software,

permiten la implementación de esta técnica, dentro de los cuales se encuentran SWISS-MODEL [79], IntFOLD [125], ROSETTA [107], MODELLER [60], entre los principales.

Por otro lado, existen diferentes métodos computacionales que permiten el estudio de interacción entre proteína y una molécula o proteína-proteína, los cuales, principalmente se enfocan en el uso de técnicas como docking o dinámicas moleculares, con el fin de estudiar los posibles residuos que participan en la interacción, evaluándose a nivel energético y midiendo el desempeño en términos de error. A su vez, técnicas basadas en simulaciones moleculares, permiten comprender la interacción en sí, y, simular el comportamiento entre la molécula de interés y la proteína. Además, métodos computacionales basados en la química cuántica, han sido utilizados para comprender fenómenos de interacción a una escala mucho más precisa. No obstante, estos, son ampliamente más costosos y su uso es limitado al estudio de un número pequeño de átomos.

Existen diferentes herramientas que permiten hacer dinámicas moleculares, tales como: NAMD [144], AMBER [41]. etc., mientras que para la interacción entre moléculas, o docking, existen AutoDock [169], RosettaDock [117], GRAMM-X [168]. Además de la suite Maestro Shrödinger [151], la cual abarca funcionalidades para las diferentes acciones propuestas.

Distintos son los enfoques pueden ser considerados en el estudio de proteínas y en el análisis de su estructura. Sin embargo, los nombrados son los principales.

1.2.3 Métodos de estudio de mutaciones

De modo general, los estudios de mutaciones se basan principalmente en el análisis de la estructura ante los cambios de residuos o la adición o eliminación de estos, evaluando los cambios mediante diferencias de energía libre, entre la proteína inicial y la mutada. Herramientas como FoldX [161], SDM [137], Auto-Mute [124], etc., permiten analizar cómo afecta una mutación en términos energéticos, basándose para ello, en el uso de funciones de energía potencial y dinámicas moleculares asociadas a dicha sustitución. Sin embargo, el uso de este tipo de herramientas, conlleva un gran costo computacional debido a los diferentes cálculos que son requeridos. Durante el capítulo 2 se ahondarán más en estas herramientas. No obstante, en la Tabla 2.7 se resumen algunas de las principales herramientas utilizadas para este tipo de análisis.

1.2.4 Métodos basados en minería de datos y aprendizaje supervisado

La minería de datos y el aprendizaje de máquinas, han sido utilizados en diferentes áreas del estudio de proteínas, ya sea para predicción de estructura secundaria [92, 129, 179],

análisis del efecto de mutaciones [36, 38, 171], identificador de patrones mediante métodos de clustering [160, 135], entre los principales ejemplos.

Diferentes enfoques han sido aplicados para obtener resultados relevantes, por un lado, se encuentra la utilización de algoritmos de aprendizaje supervisados clásicos como métodos de clasificación o predicción. por otro, el uso de métodos de clustering para identificación de patrones basados en entornos no supervisados. Actualmente, se ha empleado el uso de redes neuronales y deep learning para manipulación de sistemas de datos complejos y se han enfocado principalmente en el estudio de predicción de interacciones y evaluación de estructura secundaria de una secuencia lineal.

A pesar de los diferentes objetivos, es necesario el desarrollo de conjuntos de datos que sean descritos mediante atributos, los cuales permitan alimentar estos modelos, para generar el aprendizaje. Distintas técnicas han sido utilizadas para caracterizar los ejemplos. dentro de las cuales, principalmente se encuentran la codificación mediante One hot encoder [140], el uso de frecuencias de residuos [136] y la descripción empleando propiedades fisicoquímicas en conjunto con la caracterización del ambiente [36, 38].

Actualmente, la minería de datos y el aprendizaje automático, son una de las áreas de desarrollo de mayor interés, ya que, generan una disminución en cuanto al tiempo de cómputo y maximiza los espacios de búsqueda, los cuales, por medio de técnicas experimentales, es muy complejo analizarlas y empleando métodos computacionales para evaluar las interacciones, demandan un alto costo computacional.

Otro tipo de enfoque, se basan en el modelamiento matemático de proteínas, empleando estructuras de grafos [34, 176], con el fin de aprovechar las características y ventajas que entrega este análisis, para poder descubrir patrones o estudiar interacciones por medio de la formación de aristas entre los diferentes nodos [122].

Enfoques particulares se han desarrollado con el fin estudiar estructuras específicas o regiones de interés, ejemplos como la identificación de epítopes en secuencias lineales de antígenos, son una de las problemáticas más relevantes y de mayor impacto en los últimos años [96, 132, 155]. Sin embargo, la complejidad es alta, dada, la basta cantidad de información existente y a que las regiones con las que pueden interactuar, presentan un espacio muestral del orden del 10^9.

Los resultados satisfactorios obtenidos mediante la aplicación de técnicas de minería de datos y aprendizaje de máquinas, demuestran el poder de éstas en las diferentes áreas de estudio asociadas a la ingeniería de proteínas, convirtiéndola en una de las temáticas de mayor impacto en el último tiempo. Inclusive, permite ser un complemento relevante para investigaciones de alto impacto, tal es el caso de investigadores como Frances H. Arnold, quien en el último tiempo, ha enfocado su análisis de evoluciones dirigidas hacia un enfoque

de aprendizaje de máquinas, empleando diferentes técnicas asociadas al reconocimiento de patrones y la clasificación de estos [15, 183, 185].

1.3 Minería de datos

Minería de datos es el proceso de descubrimiento de patrones en set de datos, involucrando métodos asociados a Machine Learning [128], estadísticas y sistemas de bases de datos [82]. Se define como un subcampo interdisciplinario de la informática, el cual tiene por objetivo general extraer información (a través de métodos inteligentes) de un conjunto de datos y transformar la información en una estructura comprensible para su uso posterior [63, 58].

La minería de datos es el paso de análisis del proceso de *descubrimiento de conocimiento en bases de datos*, o KDD [62]. Además del análisis en bruto de los datos, también incluye aspectos de manipulación de bases de datos y pre procesamiento de estos, evaluaciones de modelo e inferencia, métricas de interés, consideraciones de complejidad, post procesamiento de estructuras descubiertas, visualización y actualización de la información [19].

En la Figura 1.2, se exponen las principales ramas que componen la minería de datos y los diferentes procesos que se asocian a dichas ramas.

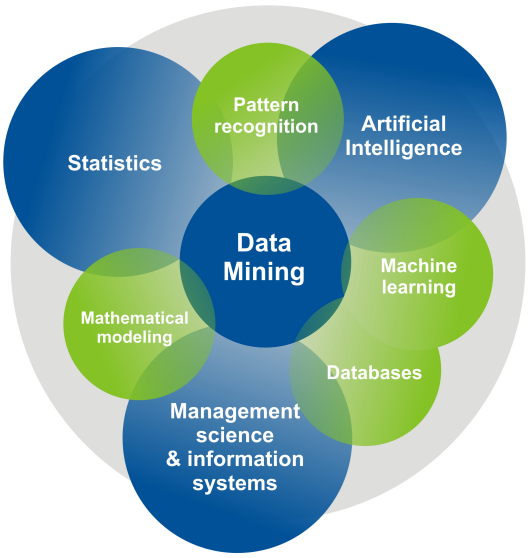


Fig. 1.2 Componentes principales de la minería de datos

Son tres las principales áreas que abarca la minería de datos: Estadística, Inteligencia Artificial y Manipulación de sistemas de información. Por otro lado, son distintos procesos los que interactúan entre estas ramas, tales como: Modelamiento Matemático, reconocimiento de patrones, Sistemas de almacenamiento persistente y machine learning [82].

Cada área en particular, tiene un objetivo general y diversos objetivos específicos. Sin embargo, estas áreas interactúan entre sí, con el fin de poder extraer patrones de información que generen conocimientos a partir de la data de procesada [19].

La minería de datos se utiliza en diferentes campos, tales como: genética y genómica [108, 149], ingeniería de proteínas [81, 110, 111], comercio y negocios [90], sistemas de tránsito [118], optimizaciones en procesos industriales [48, 71, 23], reconocimiento de patrones [94, 61], rasgos cuantificables en enfermedades [186, 131, 56] y más recientemente en áreas de dinámicas moleculares [45, 184] y parámetros para la generación de pipe lines automatizados de simulaciones cuánticas en sistemas químicos [121, 52, 148].

1.4 Principales problemáticas en la ingeniería de datos

Diferentes son las problemáticas que pueden existir en el campo de la ingeniería de proteínas, ya sea, desde la generación de herramientas computacionales para estudiar mutaciones y su efecto de manera masiva, hasta diseño de mutaciones basados en secuencias lineales de proteínas. A continuación se presentan diferentes problemáticas existentes en el área, algunas de las cuales serán motivos de estudio y desafíos a cumplir durante el presente trabajo.

1.4.1 Diferentes respuestas, una misma solución

El desarrollo de modelos de clasificación y/o regresión, es uno de los temas más recurrentes en el campo de la minería de datos y el aprendizaje de máquinas. Sin embargo, el hecho de asociar mutaciones a una respuesta, conlleva al problema de cómo caracterizarla, con el fin de alimentar a los algoritmos para ser entrenados.

A raíz de esto, cuáles son los mejores descriptores para una mutación?, desde qué puntos de vista se puede hacer una caracterización? y cuáles son más relevantes?, son interrogantes que se presentan a la hora de abordar su representación, siendo problemas que han sido tratados desde un largo tiempo, sin lograr generar un consenso o una forma general de diseñar tal representación.

En un gran número de trabajos, en los cuales se ha evaluado la estabilidad de proteínas en torno a la mutación, se han utilizado descriptores termodinámicos y de ambiente para poder representar el elemento [36, 38]. A pesar de que los desempeños de los estimadores han sido aceptables y relativamente altos. Esta caracterización ¿podrá ser utilizada para mutaciones asociadas a riesgo clínico?, ¿Existirá una correlación entre la respuesta y las variables de interés?, ¿Cómo afecta al desempeño del modelo la existencia de diferentes ejemplos

asociados a distintas proteínas en un único conjunto de datos?, etc., son interrogantes que nacen a la hora de plantearse la situación.

Dado a lo anterior, y con el objetivo de generar un aporte significativo al desarrollo de estimadores basados en aprendizaje de máquinas, se ha propuesto adicionar el concepto de filogenia a la descripción de mutaciones y disgregar los conjuntos de elementos para ser tratados por proteínas independientes, esto con el fin de generar modelos de clasificación y/o regresión proteína-específicos, los cuales puedan ser aplicados a diferentes respuestas de interés ya sea: efectos en mutaciones, estabilidad, actividad, productividad, etc., siendo éste, el tema central a abordar en el capítulo 2.

1.4.2 Codificaciones, cuáles es la mejor alternativa?

A menudo, el uso de secuencias lineales de proteínas se relaciona a la identificación de patrones o evaluación de variantes para una misma proteína. Actuales herramientas bioinformáticas permiten el uso de la secuencia de manera directa y por medio de alineamientos de secuencias o modelamiento a través del uso de Cadenas de Markov facilitan el reconocimiento de patrones o la evaluación de mutaciones. No obstante, para la aplicación de métodos basados en minería de datos, ya sea la identificación de clusters o el entrenamiento de modelos, se requiere codificar la secuencia.

Existen diferentes codificaciones posibles, ya sea, para representar la secuencia o para la caracterización de mutaciones. A pesar de ello, no existe un consenso asociado a qué técnica utilizar. Cada una presenta sus pros y contra. No obstante, la cantidad de información involucrada varía entre ellas. Sin embargo, a mayor información, incrementa el número de dimensiones a tratar, aumentando la complejidad del problema. Esto implica, utilizar técnicas de reducción de dimensionalidad para seleccionar las dimensiones con mayor variabilidad en el conjunto de datos.

Una de las codificaciones más novedosas ha sido el uso de las propiedades fisicoquímicas de los residuos y su digitalización mediante transformadas de Fourier. Esto ha permitido la identificación de residuos claves en la propiedad en estudio y soluciona el problema del efecto del ambiente de los elementos participantes.

En vista de las necesidades de desarrollo de modelos de clasificación/regresión o la identificación de residuos claves y la generación de sistemas de clustering para secuencias lineales de proteína, con el fin de apoyar al diseño de mutaciones, análisis de variantes e inclusive caracterización de secuencias, sin tener conocimiento sobre su estructura. Se propone el uso de transformadas de Fourier como método de digitalización de propiedades fisicoquímicas para el desarrollo de conjuntos de datos que permitan ser entrenados para

el desarrollo de estimadores o identificar patrones, siendo el tema central a abordar en el capítulo 3.

1.4.3 Diseñar mutaciones, un arte poco apreciado

Diseñar mutaciones de manera eficiente, con una identificación adecuada de la propiedad en estudio o funcionalidad a adicionar, sin incurrir en grandes costos económicos y de recursos, es uno de los *Santos griales* de la ingeniería de proteínas. Como se nombró previamente, son dos enfoques los que utilizan actualmente: Evolución dirigida y diseño racional de proteínas.

Ambas técnicas tienen sus ventajas y desventajas. No obstante, poseen en común una demanda en tiempo elevada y se requiere de conocimientos elevados sobre la estructura para poder diseñar las mutaciones, al menos, para el caso de diseño racional.

Enfoques computacionales han sido propuestos, con el fin de minimizar los costos económicos, contemplando evaluaciones energéticas asociadas a los residuos y cómo estos afectan a la estabilidad. No obstante, no pueden ser utilizados en secuencias lineales. Además, dejan de lado el concepto filogenético en el estudio, resultado un gap entre ambos puntos de vista. Por otro lado, métodos basados en la minería de datos, sólo se han centrado en identificación de residuos o el entrenamiento de modelos para predecir estabilidad.

A partir de lo anterior, y con el fin de generar un aporte significativo en el área de diseño, se ha considerado esta problemática como un foco central y culminante para el desarrollo de este trabajo de título, proponiendo así, la implementación de una herramienta computacional, basada en técnicas de minería de datos y aprendizaje de máquinas, que permita proponer mutaciones a un conjunto de variantes con respuesta conocida. Generando la codificación de la secuencia por medio del uso de propiedades fisicoquímicas y su respectiva digitalización a través de transformadas de Fourier, seleccionando las propiedades más relevantes por medio de la aplicación de técnicas de reducción de dimensionalidad, para así, entrenar modelos de clasificación o regresión y posterior a ello, proponer mutaciones enfocadas en un filtro, aplicando herramientas de análisis de estabilidad y propensión. Toda esta problemática, el planteamiento de la metodología y qué se utilizará para llevar a cabo, se abordará en el capítulo 4.

1.4.4 Los descartados tienen algo más que decir

En la técnica de evolución dirigida, la selección de residuos o variantes, se basa en si presentan la característica deseable o no, o si aumenta la propiedad. Si el residuo no provoca el efecto deseado, éste es descartado, ya que no cumple con el criterio de selección.

Sin embargo, es posible pensar que, combinaciones lineales de residuos pueden provocar una sinergia en alguna propiedad, generando el resultado deseado. No obstante, el estudio de dichas combinaciones, o mejor dicho, las correlaciones asociativas existentes entre mutaciones no son consideradas, ya que, sólo se seleccionan aquellos que cumplen con dicho criterio. Pero, ¿qué pasa con aquellos residuos que son descartados y que al ser mutados al mismo tiempo con otro elemento provocan el efecto deseado, e inclusive, con mejores resultados que los brindados por los seleccionados?, ¿Existe información asociada a conjuntos de mutaciones que provoquen este efecto?, ¿Será posible idear una metodología *in-silico* que permita comprender este tipo correlaciones y justificar los resultados esperados?.

Como se puede comprender, este fenómeno no ha sido explotado desde el punto de vista de minería de datos, debido principalmente, a que no existen reportes de conjuntos de datos con dichas características y esto es debido a que no ha sido un foco de estudio central. Sin embargo, se cree que es una necesidad inminente, la comprensión de estos mecanismos, ya que, aumenta el espacio de búsqueda y posterior diseño de mutaciones, en un gran número de dimensiones. Además, si bien, resultados de este estilo no han sido reportados, si, a partir de experiencias de diferentes grupos con enfoque en diseño de mutaciones y evolución dirigida, han observado que residuos no seleccionables por si solos, en combinación con otro elemento, permiten obtener la característica deseable.

A pesar de que esta problemática, no se considera dentro de los temas de estudio en sí, se plantea la discusión y se propone como un problema a ser tratado en el corto plazo, debido a las grandes implicatorias que esto puede conllevar y a las expectativas que se pueden generar al respecto, siendo de utilidad a la hora de proponer nuevas mutaciones y generar un aporte significativo en el área de ingeniería de proteínas.

1.5 Hipótesis

En base a las herramientas computacionales existentes y a los problemas expuestos previamente, además, tomando en consideración los avances en minería de datos y aprendizaje de máquinas. Se propone la siguiente hipótesis.

Es factible el uso de técnicas de minería de datos y reconocimiento de patrones para el estudio y diseño de mutaciones in-silico?, considerando tanto desarrollo de modelos de evaluación cómo herramientas computacionales que permitan proponer variantes dada la información existente.

Se plantea una hipótesis general, la cual abarca los diferentes o considera los planteamientos de problemáticas expuestos. Además, se menciona que la estructura de este proyecto

es un conjunto de metodologías independientes. Es decir, cada capítulo en sí (2, 3 y 4) son herramientas computacionales independientes y tratan de resolver una problemática de las planteadas, por lo que, cada uno presenta en sí, su hipótesis y objetivos correspondientes. No obstante, a pesar de su independencia, tienen relación profunda con el abordaje de las soluciones a partir de técnicas de minería de datos, además, que es posible combinarlas, para desarrollar una suite de librerías de apoyo a la ingeniería de proteínas.

1.6 Objetivos

Continuando con la lógica expuesta previamente, es decir, cada uno de los siguientes capítulos resuelve una de las problemáticas planteadas, y contempla en sí, una herramienta computacional por sí sola. Se plantea a continuación el objetivo general.

1.6.1 Objetivo general

Diseñar e implementar suite de herramienta computacional basada en técnicas de minería de datos, aprendizaje de máquinas y reconocimiento de patrones, enfocada en el estudio de mutaciones, que permita ser un aporte sustancial en el campo de ingeniería de proteínas.

1.6.2 Objetivos específicos

Como se expuso previamente, cada siguiente capítulo corresponde a una herramienta en sí, que formará parte de esta gran suite computacional de apoyo al estudio de mutaciones *in-silico*. Dado esto, se plantean los siguientes objetivos específicos.

1. Diseñar, implementar y testear, herramienta computacional, inspirada en la estrategia de Meta-learning, para la evaluación de mutaciones puntuales el proteínas específicas, considerando como descriptores, propiedades termodinámicas, estructurales y conceptos filogenéticos.

2. Modelar, implementar y evaluar, herramienta computacional para la codificación de secuencias lineales de proteínas, empleando digitalización de propiedades fisicoquímicas, por medio de transformadas de Fourier, la cual permita la identificación de residuos claves y la aplicación de algoritmos de aprendizaje supervisado y clustering, para el entrenamiento de modelos y el reconocimiento de patrones.

3. Diseñar, implementar y testear, herramienta computacional para el diseño de mutaciones *in-silico*, basadas en técnicas de minería de datos y reconocimiento de patrones,

enfocadas en secuencias lineales y modelos de aprendizaje supervisado, cuyos descriptores sean espectros de frecuencia basados en transformadas de Fourier y se constituya por herramientas de filtro, que aseguren estabilidad a la proteína y propensión al cambio en base a conceptos filogenéticos.

Tal como se puede observar, los aprendizajes y competencias adquiridas al cumplir el objetivo 1 y 2, se utilizan en el desarrollo del objetivo 3. Lo cual denota una especie de dependencia entre las metodologías a plantear. Sin embargo, cada uno de los objetivos, corresponde a una herramienta computacional independiente, la cual podrá ser utilizada para los fines que el usuario estime conveniente. El conjunto de éstas, se asocia al desarrollo de la suite computacional de estudio de mutaciones empleando técnicas de minería de datos, reconocimiento de patrones y aprendizaje de máquinas.

Cada uno de los siguientes capítulos, presenta su propio marco teórico, además de hipótesis y objetivos, asociados a una metodología que trata de cumplirlos. No obstante, todos enfocados en un mismo punto: desarrollo de herramientas de apoyo para el estudio de mutaciones.

Chapter 2

Modelos predictivos asociados a mutaciones puntuales en proteínas

El análisis del efecto de mutaciones puntuales en proteínas, es una de las problemáticas más estudiadas en los últimos años. Las investigaciones se enfocan principalmente, en la evaluación de cambios en la estabilidad de la proteína mediante la variación de energía libre que la mutación provoca [161, 137, 152, 138].

Diferentes modelos predictivos han sido desarrollados para poder predecir cambios de energía libre, en base a algoritmos de aprendizaje supervisado o mediante técnicas de minería de datos, y así, determinar el efecto de la mutación en set de proteínas de interés [146, 39, 30, 102, 171, 72, 37]. No obstante, en casos más específicos, se han desarrollado modelos para proteínas independientes, con el fin de asociar la mutación a un rasgo clínico, particularmente, enfocado a casos de cáncer [76, 66], cambios en termo estabilidad [167], propiedades geométricas [13], entre las principales.

Sin importar el uso o la respuesta de los modelos, es necesario construir set de datos con ejemplos etiquetados, es decir, cuya respuesta sea conocida para poder entrenar modelos basados en algoritmos de aprendizaje supervisado y así evaluar su desempeño. Los enfoques principales al desarrollo de descriptores se basan en propiedades fisicoquímicas y termodinámicas, así como también, el ambiente bajo el cual se encuentra la mutación [37], ya sea a partir de la información estructural o sólo considerando la secuencia lineal. Sin embargo, no son considerados, los componentes asociados a conceptos filogenéticos y la propensión a cambios de dicha mutación generando un gap entre ambos puntos de vista [134].

Dado a los modelos existentes y en vista a la necesidad de generar nuevos sistemas de predicción para mutaciones puntuales en proteínas, en respuesta al aumento considerable de reportes en los últimos años, se propone una nueva metodología para el diseño e implementación de modelos predictivos en mutaciones puntuales de proteínas.

Las mutaciones son descritas desde los puntos de vista estructural, termodinámico y filogenético. El desarrollo de los predictores es inspirado en el concepto de Meta Learning y es apoyado con técnicas estadísticas, tanto para la selección de modelos como para la evaluación de medidas de desempeño, entregando como resultado, un conjunto de modelos para las mutaciones puntuales reportadas, unificados en un único meta modelo.

Esta metodología será aplicada para generar estimadores en diferentes proteínas con mutaciones reportadas con respuesta conocida, como por ejemplo: evaluando las diferencias de energía libre que provoca la mutación y clasificaciones para evaluar si la sustitución de residuos aumenta o disminuye la estabilidad. A su vez, se implementarán modelos de clasificación para determinar la propensión clínica en un conjunto de mutaciones conocidas relacionados con el gen *p*VHL, responsable de la enfermedad von Hippel Lindau, con el fin de exponer la versatilidad de la metodología y los problemas relevantes a set de datos altamente no-lineales.

A continuación, se describen los principales conceptos relacionados a aprendizaje supervisado, seguido de algunas herramientas computacionales para el análisis de mutaciones y su relevancia en la de estabilidad de una proteína, continuando con la metodología propuesta, la caracterización de los diferentes set de datos a utilizar y resultados parciales obtenidos al aplicar esta metodología.

2.1 Aprendizaje de Máquinas

Aprendizaje de Máquina, es una rama de la inteligencia artificial que tiene por objetivo el desarrollo de técnicas que permitan a los computadores aprender, es decir, generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos [128]. Aplicándose en diferentes campos de investigación: motores de búsqueda [50], diagnósticos médicos [47, 1], detección de fraude en el uso de tarjetas de crédito, bioinformática [106], reconocimiento de patrones en imágenes [59] y textos [130, 4], etc.

Los algoritmos de aprendizaje pueden clasificarse en dos grandes grupos [128]:

- **Supervisados**: se cumple un rol de predicción, clasificación, asignación, etc. a un conjunto de elementos con características similares, por lo que los datos de entrada son conocidos.
- **No Supervisados**: su objetivo es agrupar en conjuntos con características similares los elementos de entrada dado los valores de estos atributos, en base a la asociación de patrones característicos que representen sus comportamientos.

A continuación se describen en forma general, los algoritmos de aprendizaje supervisados utilizados para el desarrollo de la metodología, explicando los conceptos bajo los que se basan y cómo estos entrenan y se emplean para predecir o clasificar nuevos ejemplos.

2.1.1 Algoritmos de aprendizaje supervisado

Existen diferentes algoritmos de aprendizaje supervisado, los cuales pueden ser asociados a la clasificación de un elemento o la predicción de valores, dependiendo el tipo de respuesta existente en el conjunto de datos a estudiar. En el caso de respuestas con distribución continua, se trabajan con algoritmos de regresión, mientras que si la respuesta es binaria o multiclase y es representada por variables categóricas, los algoritmos se basan en clasificadores [128].

A su vez, también se pueden dividir con respecto a la forma en que se trata el problema, existiendo algoritmos basados en cálculos de distancia entre ejemplos (K-Vecinos Cercanos), otros que consideran transformaciones vectoriales y aplicaciones de funciones de kernel (Máquina Soporte de Vectores), así como también el uso de las características como entorno espacial de decisión (Árboles y métodos de ensamble) y aquellos que utilizan redes neuronales y trabajan en torno a cajas negras, o métodos basados en regresiones lineales, sólo aplicados a modelos predictivos de variables continuas.

Cada uno de estos algoritmos es descrito a continuación, enfocándose tanto en el componente matemático asociado, así como también en las ventajas y usos posibles que estos puedan tener, con respecto al conjunto de datos a trabajar.

2.1.2 Métodos basados en regresiones lineales

Regresión lineal, es uno de los métodos más simples en cuanto a predicción de variables continuas, además de uno de los más limitantes debido al sobreajuste que éste puede generar. No obstante, permite evaluar de manera simple y sencilla conjuntos de datos [78].

Matemáticamente, se espera que el conjunto de respuesta sea el resultado de una combinación lineal de parámetros, es decir. Sea $\hat{y}$ el vector de predicciones, se tiene que:

$$\hat{y}(w,x) = w_0 + w_1 x_1 + ... + w_p x_p$$

Donde w_0 es el intercepto y $w = (w_1, \cdots, w_p)$ el vector de coeficientes.

Existente diferentes métodos de regresión lineal, los cuales cumplen con el mismo objetivo. Sin embargo, la forma en la que minimizan el error asociado a las diferencias entre los valores predichos y los observados.

A modo de ejemplo, en la Tabla 2.1 se exponen distintas formas de ajustar o minimizar el error asociado.

Tipo de regresión	Minimización	Descripción
Ordinary Least Squares [75]	$\min_w \|\|Xw-y\|\|_2^2$	Método más simple, no implica parámetros externos.
Ridge Regression [89]	$\min_w \|\|Xw-y\|\|_2^2+\alpha\|\|w\|\|_2^2$	Adición de penalización α.
Lasso [83]	$\min_w \frac{1}{2n_{\text{samples}}}\|\|Xw-y\|\|_2^2+\alpha\|\|w\|\|_1$	Reduce el número de características bajo las cuales depende la solución final.
Elastic-Net [189]	$\min_w \frac{1}{2n_{\text{samples}}}\|\|Xw-y\|\|_2^2+\alpha\rho\|\|w\|\|_1+\frac{\alpha(1-\rho)}{2}\|\|w\|\|_2^2$	Usado principalmente en caso de atributos con alta correlación.
Orthogonal Matching Pursuit (OMP) [139]	$\arg\min_\gamma \|\|y-X\gamma\|\|_2^2$ subject to $\|\|\gamma\|\|_0 \leq n_{\text{nonzero\_coefs}}$	Permite fijar el número de coeficientes no nulos.
Bayesian Regression [42]	$p(y\|X,w,\alpha)=\mathcal{N}(y\|Xw,\alpha)$	Se adapta a de manera eficiente a los datos de entrenamiento y permite incluir penalizaciones asociadas a los coeficientes

Table 2.1 Tipos de regresión con su función de minimización y descripciones correspondientes a las características que estos poseen.

Pese a su simplicidad, los métodos basados en regresiones lineales han sido ampliamente utilizados. No obstante, presentan diferentes problemas asociados al sobreajuste de parámetros.

2.1.3 K-Vecinos Cercanos

Algoritmo de aprendizaje supervisado, el cual tiene por objetivo asociar un elemento a una clase en particular, dada la información de ejemplos de entrada que tengan asociadas características particulares, que puedan declararse como *vecinos* del nuevo ejemplo a clasificar, siendo **k** el número de vecinos que se está dispuesto a utilizar para aplicar la clasificación

[100]. La mejor elección de **k** depende fundamentalmente de los datos; generalmente, valores grandes de **k** reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas.

Con el fin de evaluar la cercanía de los ejemplos existentes contra el nuevo ejemplo a clasificar, es necesario asociar ciertas medidas de distancia que permitan cuantificar esta característica, para así poder comparar esta distancia y evaluar la cercanía para asociarle una clase a este nuevo ejemplo [57]. La distancia a emplear para evaluar la cercanía puede ser: Euclidiana [53], Manhattan [142], coseno [112] o Mahalanobis [119], entre las principales, las cuales son expuestas de manera general en la Tabla 2.2.

Distancia	**Fórmula**	**Descripción**
Euclideana	$D_{(X,Y)} = \sqrt{\sum_{i=1}^{l}(X_i - Y_j)^2}$	Se basa en una recta entre dos puntos
Coseno	$D_{(X,Y)} = \arccos(\frac{X^T Y}{\|\|X\|\| \, \|\|Y\|\|})$	Se basa en vectores y en el coseno del ángulo que forman
Manhattan	$D_{(X,Y)} = \sum_{i=1} n\|X_i - Y_i\|$	Distancia en forma de zig-zag
Mahalanobis	$D_{(X,Y)} = \sqrt{(X-Y)^T S^{-1}(X-Y)}$	Considera las correlaciones entre las variables de estudio

Table 2.2 Resumen tipos de distancias utilizadas en procesos de comparación de ejemplos

K-Nearest Neighbors (KNN por su descripción en inglés), presenta algunos problemas, tales como: posibles errores al existir más de un elemento de distinta clase cercano al nuevo ejemplo a clasificar. Sin embargo, dicho error estimado es reducido [100].

Existen dos variaciones para la aplicación de KNN: aplicación basada en las distancias y aplicación basada en radios con respecto a puntos, la primera es mayormente usada. No obstante, en el caso de que los puntos no se encuentren uniformemente distribuidos es una mejor opción usar la segunda alternativa, siendo muy eficaz en problemas conocidos como *la maldición de la dimensionalidad*[1].

KNN utiliza el componente de peso [165], es decir, valores asignados a puntos específicos para determinar si un elemento a clasificar es de una clase o no, normalmente se utilizan pesos uniformes. Sin embargo, es posible asignar valores de tal manera que al momento de realizar la votación puntos más cercanos en base a distancias presenten más peso que otros.

Se han implementando diversos algoritmos a la hora de aplicar la técnica de KNN, los cuales tienen relación con el coste computacional que presentan, dentro de estos se encuentran: Brute Force, K-D Tree y Ball Tree [140].

[1] Para más detalles ver sección 2.1.11

Este algoritmo de aprendizaje supervisado, puede ser utilizado tanto para el entrenamiento de modelos de clasificación (respuestas categóricas) y de regresión (respuestas continuas).

2.1.4 Naive Bayes

Naive Bayes es un conjunto de algoritmos de aprendizaje supervisados basados en la aplicación del teorema de Bayes con la suposición "ingenua" de independencia entre cada par de características [187]. Dada una variable de clase y y un vector de característica dependientes de la forma $x_1, ..., x_n$, el teorema de Bayes establece la siguiente relación:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n|y)}{P(x_1, \ldots, x_n)}$$

Utilizando la suposición ingenua de independencia de características, se tiene que:

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

Para todo i, esta relación se simplifica a:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

Dado que $P(x_1, \ldots, x_n)$ es constante dada la entrada, se puede utilizar la siguiente regla de clasificación:

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y),$$

A pesar de sus supuestos aparentemente simplificados, los clasificadores de Naive Bayes han funcionado bastante bien en muchas situaciones del mundo real, la famosa clasificación de documentos y el filtrado de spam son ejemplos de ello [109, 46, 127]. Requieren una pequeña cantidad de datos de entrenamiento para estimar los parámetros necesarios. Pueden ser extremadamente rápido en comparación con métodos más sofisticados. El desacoplamiento de las distribuciones de las características condicionales de clase significa que cada distribución se puede estimar de forma independiente como una distribución unidimensional. Esto a su vez ayuda a aliviar los problemas derivados de la dimensionalidad.

Existen distintos tipos de clasificadores de Naive Bayes, diferenciándose entre sí en la función de distribución de probabilidad que utilizan [127, 97, 120], dentro de los que se encuentran:

- **Gaussian Naive Bayes.**

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

- **Multinomial Naive Bayes.**

  La distribución se parametriza por el vector $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ para cada clase y, donde n es el número de características y θ_{y1} es la probabilidad $P(x_i \mid y)$ de que la característica i aparezca en una muestra que pertence a la clase y.

  Cada θ_y es estimado por:

$$\hat{\theta}_{yi} = \frac{N_{yi}+\alpha}{N_y+\alpha n}$$

  Donde $N_{yi} = \sum_{x \in T} x_i$ es el número de veces que aparece la característica i en la muestra de clase y en el set de entrenamiento T y $N_y = \sum_{i=1}^{|T|} N_{yi}$ representa el total de todas las características para la clase.

- **Bernoulli Naive Bayes.**

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

2.1.5 Árboles de Decisión

Se define árbol de decisión como un modelo de predicción, utilizado en el ámbito de la inteligencia artificial, en el cual, dado un conjunto de datos, se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema.

Aprendizaje basado en árboles de decisión es un método comúnmente utilizado en la minería de datos, cuyo objetivo consiste en desarrollar un modelo de predicción para el valor de una variable de destino en función de diversas variables de entrada [67].

El aprendizaje basado en árboles de decisión utiliza un árbol como un modelo predictivo que mapea las observaciones de las características que presenta un elemento. En estas estructuras de árbol, las hojas representan etiquetas de conjuntos ya clasificados, los nodos, a su vez, nombres o identificadores de los atributos y las ramas representan posibles valores para dichos atributos [20].

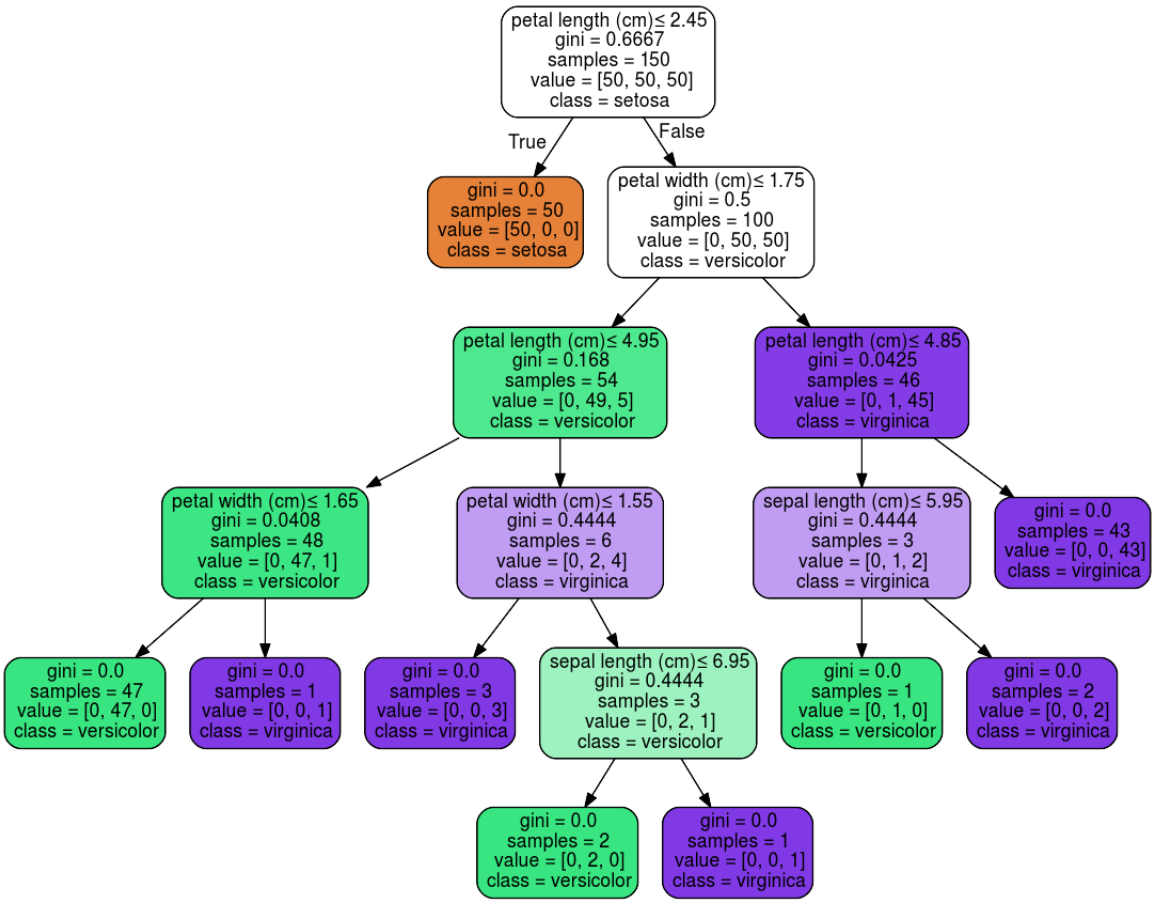


Fig. 2.1 Estructura de árbol para modelo de clasificación de set de datos iris.

A modo de ejemplo, se expone en la Figura 2.1, una representación de un posible árbol, el cual fue desarrollado para entrenar modelos de clasificación utilizando el set de datos iris [65].

Un árbol puede ser entrenado mediante el fraccionamiento del conjunto inicial en subconjuntos basados en una prueba de valor de atributo. Este proceso se repite en cada subconjunto derivado de una manera recursiva, el cual es denominado *particionamiento recursivo*. La recursividad termina cuando el subconjunto en un nodo m tiene todos sus miembros el mismo valor de la variable categórica o pertenecen al mismo intervalo finito, para el caso en que las respuestas se asocian a distribuciones continuas, o cuando la partición ya no agrega valor a las predicciones.

Para cada división, es necesario el uso de una función que entregue una medida de impureza en cada partición, esto, con el objetivo de seleccionar el mejor subconjunto para un atributo dado, la elección de dicho descriptor, se basa en el objetivo de separar de mejor manera los ejemplos.

La selección de los atributos se basa en qué atributo, al momento de clasificar, genera nodos más puros. Para ello, se utiliza una función de ganancia de información, la cual representa la ganancia obtenida a partir de una división de los ejemplos de entrenamiento [28].

Existen diferentes tipos de algoritmos bajo los cuales son implementados las ideologías de los árboles de decisión, cada uno presenta características particulares. Sin embargo, cumplen con el mismo objetivo. A modo general, en la Tabla 2.3 se exponen algunos de los algoritmos existentes [91].

Algoritmos de árboles de decisión	
Algoritmo	**Descripción**
Iterative Dichotomiser 3 (ID3)	Se crea un árbol de múltiples vías, encontrando para cada nodo la característica categórica que producirá la mayor ganancia de información. Por lo general, se aplica un paso de poda para mejorar la capacidad del árbol asociada a la generalización.
C4.5	Permite manipular variables continuas, mediante la definición dinámica de atributos discretos, dividiendo el valor continuo en un conjunto finito de intervalos discretos. La poda se realiza eliminando la condición previa de una regla si la precisión de la regla mejora sin ella.
C5.0	Genera un conjunto de reglas más pequeñas en comparación a C4.5. Sin embargo, este último es más preciso en cuanto al entrenamiento.
Classification and Regression Trees (CART)	Construye árboles binarios utilizando la característica y el umbral que producen la mayor ganancia de información en cada nodo.

Table 2.3 Resumen de algoritmos comunes para la creación de árboles de decisión.

Formulación matemática

Una definición matemática, tanto del proceso de clasificación o regresión y cómo son los criterios de selección de atributos es expuesta a continuación.

Sea $x_i \in R^n$ los vectores de entrenamiento del conjunto de datos y sea $y \in R^l$ el vector de respuestas asociadas a cada ejemplo. Un árbol de decisión divide el espacio de forma recursiva, de manera que las muestras con las mismas etiquetas se agrupan.

Cada nodo m puede ser representado por Q y sea $\theta = (j, t_m)$ la división candidata para un atributo j y un umbral t_m, se definen las particiones $Q_{left}(\theta)$ y $Q_{rigth}(\theta)$ tal que:

$$\begin{aligned} Q_{left}(\theta) &= (x,y)|x_j <= t_m \\ Q_{right}(\theta) &= Q \setminus Q_{left}(\theta) \end{aligned} \tag{2.1}$$

Asociado a las divisiones, se tiene que, cada nodo generado se mide con respecto a la impureza de éste, la cual, puede ser representada por una función $H()$ y a la ganancia de información que genera la división $G(Q,\theta)$, la cual se estima como:

$$G(Q,\theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)$$

Los descriptores se seleccionan con respecto a aquel que minimice la impureza de los nodos:

$$\theta^* = \text{argmin}_\theta \, G(Q,\theta)$$

Finalmente, se tiene que para los subconjuntos $Q_{left}(\theta^*)$ y $Q_{rigth}(\theta^*)$ la profundidad máxima se alcanza si:

- $N_m < min_{samples}$
- $N_m = 1$

Con N_m representando el número de nodos m.

Los criterios de clasificación se basan en la proporción de las clases según sus observaciones y en la función de impureza que es posible utilizar.

Si el vector de respuestas, es asociado a variables categóricas y toma valores entre $0, 1, \cdots, k-1$ para un nodo m, representando una región R_m con N_m ejemplos, se tiene que la proporción de observaciones de clase k en un nodo m puede definirse como:

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k)$$

Con respecto a las diferentes funciones de impureza $H()$ que pueden ser utilizadas se tienen las siguientes, descritas en la Tabla 2.4.

Función	**Fórmula**
Gini	$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$
Entropía	$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$
Misclassification	$H(X_m) = 1 - \max(p_{mk})$

Table 2.4 Tipos de medidas de impureza que pueden ser utilizadas en árboles de decisión para modelos de clasificación.

Siendo X_m datos de entrenamiento en el nodo m.

A la hora de entrenar modelos de regresión, es decir, con respuestas asociadas a una distribución continua, se tiene que para el nodo m, el cual representa una región R_m con observaciones N_m, los criterios comunes para minimizar errores en futuras divisiones son el Error cuadrático medio y el Error absoluto medio, quienes minimizan el error tipo II y el error tipo I, respectivamente.

Estos se pueden definir como:

- **Error cuadrático medio**:

$$\begin{aligned} \bar{y}_m &= \frac{1}{N_m} \sum_{i \in N_m} y_i \\ H(X_m) &= \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2 \end{aligned} \tag{2.2}$$

- **Error absoluto medio**:

$$\begin{aligned} \bar{y}_m &= \frac{1}{N_m} \sum_{i \in N_m} y_i \\ H(X_m) &= \frac{1}{N_m} \sum_{i \in N_m} |y_i - \bar{y}_m| \end{aligned} \tag{2.3}$$

Este tipo de entrenamiento, es uno de los más utilizados, debido a su simplicidad a la forma en la que trabaja, ya que, permite comprender del problema, con respecto a los atributos y cómo estos van distribuyendo las respuestas, así, es posible entender las decisiones que toma el algoritmo para clasificar o predecir nuevos ejemplos, determinar comportamientos preferentes y tendencias sobre atributos y rangos de estos.

2.1.6 Support Vector Machine (SVM)

Máquina soporte de vectores (SVM por sus siglas en inglés), es un conjunto de métodos de aprendizaje supervisado, utilizados para clasificar, predecir e inclusive para la detección de puntos outliers [159].

SVM genera una representación de los ejemplos como puntos en el espacio, mapeados de modo que los ejemplos de las categorías separadas se dividan por un espacio claro que es tan

amplio como sea posible. Nuevos ejemplos son entonces mapeados en ese mismo espacio y predicen si pertenecen a una categoría en base a qué lado del espacio son asignados [159].

Las predicciones se realizan de manera eficiente, utilizando funciones kernel para su transformación en espacios no lineales [6]. Esto permite generar transformaciones de espacio dimensional de los datos, para mapear implícitamente sus entradas en espacios característicos de alta dimensión.

Las funciones de kernel que pueden ser utilizadas, son descritas en la Tabla 2.5

Kernel	**Fórmula**
Lineal	$\langle x, x' \rangle$
Polinomial	$(\gamma \langle x, x' \rangle + r)^d$
Radio basis function (RBF)	$\exp(-\gamma \|x - x'\|^2)$
Sigmoideo	$\tanh(\gamma \langle x, x' \rangle + r)$

Table 2.5 Tipos de kernels aplicados por SVM para la transformación espacial de los datos.

Existen ventajas y desventajas al usar SVM como algoritmo de aprendizaje supervisado, las cuales se listan a continuación.

- Efectivo en espacios de alta dimensión.
- Eficiente cuando el número de atributos supera a la cantidad de ejemplos en un conjunto de datos.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamada vectores de soporte), por lo que también es memoria eficiente.
- Versátil: diferentes funciones de kernel pueden ser especificadas para la función de decisión.

Las desventajas de las máquinas de soporte vectorial incluyen:

- Si el número de características es mucho mayor que el número de muestras, es probable que el método tenga un mal desempeño.
- SVM no proporciona directamente estimaciones de probabilidad [182], estos se calculan utilizando cinco veces una costosa validación cruzada.

Existen diversas variaciones de SVM, tales como: SVC [80], νSVC y LinearSVC, los cuales son capaces de realizar una clasificación multiclase[2] en un conjunto de datos, es

[2]Implica la existencia de un número de clases mayor a dos

decir, ya no depender de un clasificador binario. A su vez, presentan sus variantes para el entrenamiento de modelos de regresión, asociados a respuestas con distribución continua, denominados SVR, νSVR y LinearSVR, respectivamente.

De manera simple, se puede describir algunos puntos como:

- SVC es implementado basado en libsvm [43]. La complejidad del tiempo de ajuste se hace cuadrática con el número de muestras, lo que dificulta escalar a conjunto de datos con tamaño mayor a 10000 ejemplos.
- Por otro lado, νSVC presenta características similares a SVC. Pero, utiliza el parámetro ν para controlar el número de vectores de soporte. Su implementación, al igual que SVC se basa en libsvm.
- LinearSVC es similar a SVC pero, se utiliza una función de kernel lineal. Además, es implementado en términos de liblinear, por lo que tiene más flexibilidad en la elección de las penalizaciones y las funciones de pérdida y facilita una mayor escalabilidad para conjuntos de datos con gran cantidad de ejemplos.

Cada uno de los clasificadores expuestos en los puntos anteriores toman como entrada el set de entramiento y las etiquetas asociadas a las clases, con el fin de generar tanto el testeo como la validación del modelo.

Previo a la etapa de entrenamiento, se utilizan vectores de apoyo para el set de entrenamiento, los que son denominados vectores de soporte, normalmente obtenidos a partir de funciones de kernel para.

SVC y NuSVC implementan el enfoque *uno contra uno* para la clasificación multiclase. Si existen n clases, se construyen $\frac{n*(n-1)}{2}$ clasificadores, de los cuales cada uno forma un set datos de dos clases; por otro lado, LinearSVC implementa una estrategia multi-clase *uno contra el resto*, formando así modelos de n clases, los cuales son entrenados n veces. Si sólo hay dos clases, sólo se entrena un modelo.

Los algoritmos SVM están asociados a diversos problemas. Sin embargo, el principal, radica en el desbalance de clases, ya sea por el número que presentan o por el peso asociado a éstas, tal como se expone en la Figura 2.2:

Formulación matemática

SVM construye un hiperplano o un conjunto de hiperplanos en un espacio altamente dimensiona, el cual es utilizado tanto para la clasificación de elementos, com para la predicción de valores continuos, a su vez, puede ser implementado para detección de puntos outliers.

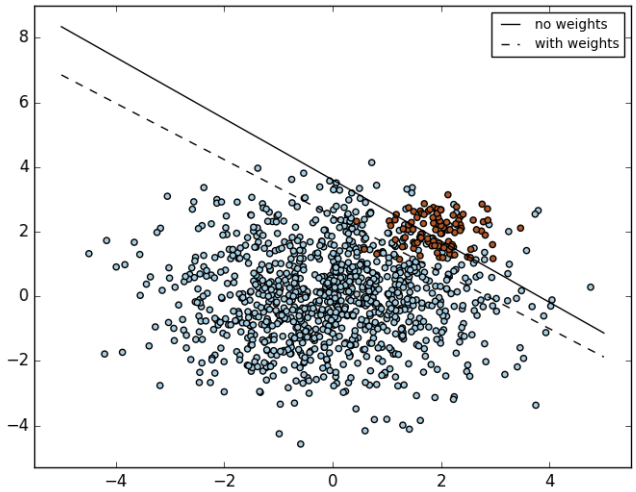


Fig. 2.2 Muestra de desbalance de clases en SVM.

Intuitivamente, se logra una buena separación por el hiper plano que tiene la mayor distancia a los puntos de datos de entrenamiento más cercanos de cualquier clase, ya que en general, cuanto mayor sea la separación, menor será el error de generalización del modelo.

Es posible observar este comportamiento en la Figura 2.3

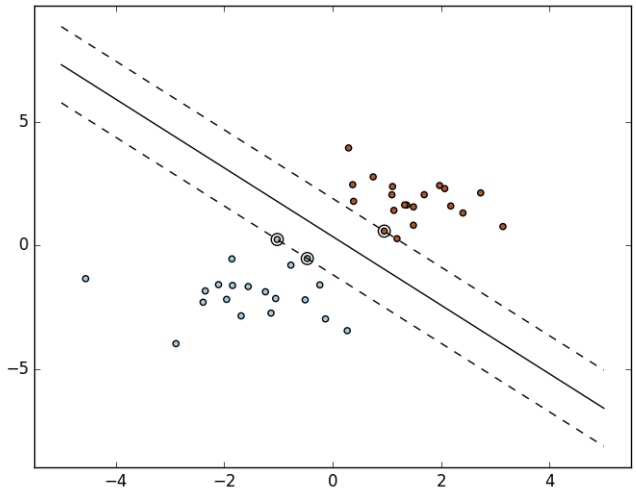


Fig. 2.3 Esquema de hiperplanos en SVM.

Es posible entregar una definición matemática para SVM, haciendo una diferenciación asociados a los métodos de clasificación y regresión. De tal manera que, para SVC, se tiene que:

Dado los vectores de entrenamiento $x_i \in R$, i=1,..., n, en dos clases, y un vector, $y \in \{1,-1\}^n$, SVC resuelve el siguiente problema primario:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$para\ la\ clase\ y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,\ \zeta_i \geq 0, i = 1,...,n$$

Su doble es

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

$$para\ la\ clase\ y^T \alpha = 0\ 0 \leq \alpha_i \leq C, i = 1,...,n$$

Donde e es el vector de todos los unos, $C > 0$ es el límite superior, Q es una matriz de $n\ x\ n$ definida semipositiva, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, donde $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ es el kernel. Los vectores de entrenamiento son implícitamente mapeados en un espacio dimensional mayor (tal vez infinito) por la función ϕ.

La función de decisión es:

$$sgn(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

Por otro lado, νSVC presenta características similares a SVC. No obstante, el uso del parámetro ν permite controlar el número de vectores de soporte y errores de entrenamiento. $\nu \in (0,1]$ y es un límite superior en la fracción de errores de entrenamiento y un límite inferior de la fracción de vectores de soporte.

A su vez, LinearSVC presenta el mismo comportamiento a SVC. Sin embargo, la gran diferencia es la utilización de un kernel lineal como función de transformación de espacio y evaluación de puntos.

Finalmente, ya que SVM también permite entrenar modelos de regresión particularmente con SVR, una definición matemática de éste se expone como:

Dados los vectores de entrenamiento $x_i \in R^n$ $\varepsilon - SVR$ [162] resuelve el siguiente problema primario:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$

$$para\ la\ clase\ y_i - w^T \phi(x_i) - b \leq \varepsilon + \zeta_i,$$

$$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*,\ \zeta_i, \zeta_i^* \geq 0, i = 1, ..., n$$

Donde e es el vector para todos, $C > 0$ es el límite superior, Q es una matriz de n x n definida semipositiva, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ es el kernel. Aquí los vectores de entrenamiento son implícitamente mapeados en un espacio dimensional mayor (tal vez infinito) por la función ϕ.

La función de decisión es:

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + \rho$$

Como se puede observar, las formulaciones matemáticas, tanto para SVC como SVR son similares. Sin embargo, se diferencian en la función de decisión a utilizar, esto es debido a que SVC es un clasificador, mientras que SVR es un métodos de regresión.

2.1.7 Métodos de ensamble

Los métodos de ensamble, se basan en la combinación de las predicciones obtenidas por varios estimadores, construidos en base a algoritmos de aprendizaje supervisado, con el fin de mejorar la generalización del modelo y aumentar la robustez ante nuevos ejemplos [55].

Existen dos familias de métodos de ensamble, las cuales se diferencian principalmente en la forma en que combinan los modelos para obtener la medida de desempeño final [104]:

1. **Métodos ponderados**: basados en la construcción de varios estimadores independientes y promediar sus medidas de desempeño, esto mejora el rendimiento debido a que disminuye la variabilidad de las clasificaciones. Ejemplos comunes de esto son Bagging y Random Forest.

2. **Métodos boosting**: basados en la construcción secuencial de modelos, intentando disminuir el sesgo del modelo combinando diferentes estimadores débiles. Cumple con la filosofía *"la unión de varios modelos débiles, puede construir uno fuerte"*. Ejemplos comunes de esto son AdaBoost y Gradient Tree Boosting.

A continuación, se explican brevemente algunos de los algoritmos asociados a la familia de métodos de ensamble.

Bagging

Bagging forma parte de los métodos ponderados, en particular, se puede definir como métodos que forman una clase de algoritmos compuestos por varias instancias de un estimador,

entrenados en base a subconjuntos aleatorios del set de datos original, ponderando sus predicciones individuales en una respuesta ponderada. El objetivo general de estos métodos es reducir la varianza de un estimador, por medio del proceso de entrenamiento de subconjuntos aleatorios [24].

Existen diferentes formas de generar los subconjuntos aleatorios de entrenamiento, dentro de las cuales se destacan las siguientes.

- Los subconjuntos aleatorios del conjunto de datos se basan en subconjuntos aleatorios de las muestras, esto se conoce como *"Pasting o Pegado"* [26].
- Las muestras se extraen con reemplazo, siendo este método conocido como *"Bagging"* [24].
- Los subconjuntos aleatorios del conjunto de datos se basan en subconjuntos aleatorios de las características, esto se conoce como *"subespacios aleatorios"* [12].
- Los subconjuntos aleatorios se crean en base a subconjuntos aleatorios de características y muestras, esto se conoce como *"Random Patches"* [114].

Random Forest

Random Forest es un método de ensamble ponderado basado en árboles de decisión aleatorios. Conjuntos de diversos clasificadores son creados basados en efectos aleatorios tanto de la extracción de características como de ejemplos, formando subconjuntos de elementos, cada uno de estos aporta con un valor de estimación, el cual es ponderado con los restantes, obteniendo así, la medida general [25].

Un esquema representativo del proceso, es como se expone en la Figura 2.4. En ella, se aprecian que se generan *n* árboles, los cuales contemplan diferentes cantidades de ejemplos o atributos y la estimación final se basa en una ponderación, ya sea por proceso de votación, en el caso de modelos de clasificación, o simplemente por la media, para modelos de regresión [27].

AdaBoost

AdaBoost, es un algoritmo basado en el método boosting, lo que implica que se ajusta a una secuencia de estimadores débiles obtenidos a partir de diferentes subconjuntos de datos generados de manera aleatoria desde el conjunto inicial de datos de entrenamiento [35].

Cada una de las predicciones obtenidas por los estimadores se combinan de manera ponderada por votación, en el caso de modelos de clasificación, o a través de un promedio en base a las estimaciones resultantes, en el caso de modelos de regresión.

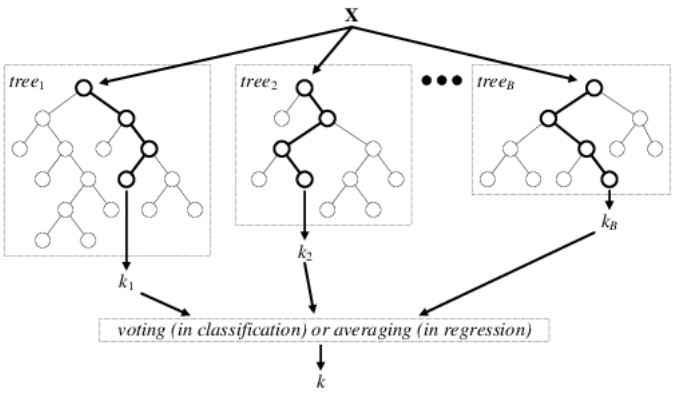


Fig. 2.4 Esquema representativo de algoritmo Random Forest.

Las modificaciones de los datos en cada iteración de boosting, consisten en aplicar pesos $w_1, w_2, .., w_N$, a cada una de las muestras de entrenamiento.

Inicialmente, todos los pesos están configurados en $\Psi_i = 1/N$, de modo que el primer paso simplemente entrena un modelo débil en los datos originales. Para cada iteración sucesiva, las ponderaciones de la muestra se modifican individualmente y el algoritmo de aprendizaje se vuelve a aplicar a los datos ponderados.

En un paso dado, los ejemplos de entrenamiento que fueron predichos incorrectamente por el modelo mejorado inducido en el paso anterior tienen sus pesos incrementados, mientras que los pesos se disminuyen para aquellos que fueron predichos correctamente [84].

A medida que avanzan las iteraciones, los ejemplos que son difíciles de predecir reciben una influencia cada vez mayor. Por lo tanto, cada modelo de aprendizaje débil subsiguiente se ve forzado a concentrarse en los ejemplos que se pierden en los anteriores en la secuencia.

Gradient Tree Boosting

Gradient Tree Boosting o Gradient Boosted Regression Trees, es una generalización de métodos de boosting para funciones diferenciables arbitrarias de pérdida [68]. Es un método considerado como preciso y efectivo, el cual puede usarse tanto para el desarrollo de modelos de clasificación como de regresión, siendo usado en diferentes áreas de investigación: motores de búsqueda, ecología, minerología, biotecnología, entre otros.

Dentro de las principales ventajas que posee, se encuentran: manejo natural de diferentes tipos de características en un set de datos, alto poder predictivo y robusto frente a la predicción de valores atípicos en una muestra [69].

Como formulación matemática, se tiene, la adición de modelos se basa en:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x)$$

Donde $h_m(x)$ es una función básica, conocida como *weak learners* en el contexto de boosting. En particular, Gradient Tree Boosting, utiliza árboles de decisión de tamaño fijo como *weak learners*.

Los modelos aditivos, se construyen de manera incremental, tal que:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

De tal manera que el nuevo árbol agregado h_m trata de minimizar la pérdida L dado el previo ensamble F_{m-1}, es decir:

$$h_m = \arg\min_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i))$$

Este algoritmo, resuelve el problema de minimización numéricamente, a través de steepest descent [14]. La dirección de steepest descent es el gradiente negativo de la función de pérdida L evaluada en el modelo actual F_{m-1}, el cual puede ser calculado para cualquier función de pérdida diferenciable, tal que:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_F L(y_i, F_{m-1}(x_i))$$

Donde la longitud del paso γ_m es seleccionada mediante una búsqueda lineal, aplicando:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)})$$

A la hora de estimar un valor continuo por medio de algoritmos de regresión o clasificar un nuevo ejemplo a través de un clasificador, existe una única diferencia, la cual radica en la función de pérdida L que es utilizada. En la Tabla 2.6 se resumen las funciones de pérdida y sus definiciones, normalmente utilizadas en métodos basados en Gradient Tree Boosting.

Funciones de pérdida utilizadas en métodos GTB		
Función	**Descripción**	**Uso**
Least squares	Valor inicial se obtiene a partir de la media de los vectores respuesta	Regresión
Least absolute deviation	Valor inicial se obtiene a partir de la mediana de los vectores respuesta	Regresión
Huber	Combina el uso de mínimos cuadrados y los errores absolutos	Regresión
Quantile	Se basa en el uso de cuantiles para crear rangos de predicción	Regresión

Binomial deviance	Se basa en la distribución de probabilidad binomial, para modelos binarios de clasificación	Clasificación
Multinomial deviance	Se basa en la distribución de probabilidad multinomial y su valor inicial corresponde a las probabilidades a priori de cada categoría	Clasificación
Exponential loss	Se basa en distribución exponencial y sólo puede ser utilizada en modelos de clasificación binarios	Clasificación

Table 2.6 Funciones de pérdida comunes utilizadas en los algoritmos Gradient Tree Boosting, ya sea en forma de clasificador como regresor.

2.1.8 Redes Neuronales y Deep Learning

Redes neuronales es posible definirlas como una serie de modelos de aprendizaje que se basan en la forma de trabajo de las redes neuronales biológicas, es decir, se usa el concepto de *neurona* para estimar una función aproximada, la cual dependerá de un largo número de inputs, generalmente desconocidos.

En la Figura 2.5 se aprecia un sistema de red neuronal, en la cual se observa un sistema interconectado por neuronas, las cuales intercambian información en forma de mensaje entre ellas, además cada interconexión tiene un peso, el cual es un valor numérico, que puede ser obtenido en base a la experiencia.

En resumen, una red neuronal es un conjunto de entradas y salidas regidas por capas intermedias que permiten evaluar la salida, dichas capas operan entre sí en base a funciones matemáticas y brindan un peso a la conexión, finalmente cada capa es usada para diseñar un modelo de aprendizaje supervisado o no.

Deep Learning es una herramienta de Machine Learning la cual tiene por objetivo modelar abstracciones de alto nivel en los datos por medio del uso de múltiples capas de procesamiento, ya sea a través del uso de estructuras complejas a través de múltiples transformaciones no lineales [17, 16, 54].

La investigación en esta área tiene por objetivo generar mejores representaciones y crear modelos para aprender de éstas a partir de datos no marcados a gran escala. En geneal, las representaciones obtenidas se inspiran en los avances en la neurociencia y se basa libremente en la interpretación de los patrones de procesamiento y comunicación de información en un

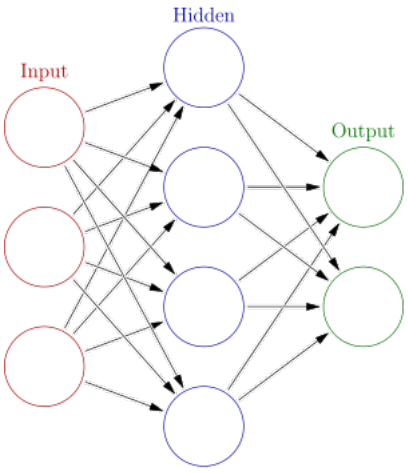


Fig. 2.5 Representación esquemática de una Red Neuronal

sistema nervioso, como la codificación neural que intenta definir una relación entre varios estímulos y respuestas neuronales asociadas en el cerebro [16].

Deep learning es un método específico de machine learning el cual incorpora redes neuronales organizadas en capas consecutivas para poder aprender iterativamente utilizando un conjunto de datos. Deep learning es especialmente útil cuando se desea aprender patrones provenientes de datos no estructurados [54].

Posee diversas arquitecturas, tales como: deep learning network, matrices de convoluciones, redes neuronales recurrentes, etc. las cuales han sido utilizadas en visión artificial para el reconocimiento de patrones, aprendizaje de escritura, etc. Deep Learning es una herramienta de Machine Learning la cual tiene por objetivo modelar abstracciones alto nivel en los datos por medio del uso de múltiples capas de procesamiento, ya sea a través del uso de estructuras complejas a través de múltiples transformaciones no lineales [9].

Dentro de los principales algoritmos que son utilizados en redes neuronales se encuentran Back Propagation [86] y Multi Layer Perceptron [153].

Si bien en la actualidad, redes neuronales y Deep Learning son metodologías ampliamente utilizadas y han tenido resultados satisfactorios a la hora de trabajar en diferentes áreas de investigación, presentan un problema relevante al momento de aprender de los datos, los atributos y cómo estos facilitan o distribuyen la información.

Este problema es debido a que, los sistemas de redes neuronales trabajan en torno a información oculta, denominados, sistemas de cajas negras, lo cual, no permite comprender cómo se genera una nueva clasificación o predicción de elementos.

Lo anterior, no ocurre con métodos basados en estimaciones de distancia como KNN, uso de hiperplanos y funciones de kernel para la transformación espacial de los atributos

como SVM, reglas de decisión que permiten representar estructuras de árbol que facilitan la comprensión de cómo distribuyen los atributos, rangos preferibles etc., como lo hacen los algoritmos basados en árboles de decisión e inclusive los métodos de ensamble. Es decir, diferentes algoritmos vislumbran cómo manipulan la información para llegar al resultado, no siendo el caso de las redes neuronales. Por lo tanto, dado a que se requiere de algoritmos que permitan la comprensión del problema y que posibiliten generar aprendizaje de los atributos, se descartan a priori el uso de métodos basados en redes neuronales y el uso de Deep Learning.

2.1.9 Meta-Learning

El Meta-Learning, es un campo relacionado a aprendizaje supervisado, inspirado en el concepto *"aprender a aprender"* [157], se basa en el uso de meta datos y presenta el objetivo de mejorar las medidas de desempeño de un clasificador, guiándolo entorno al aprendizaje. Por otro lados, estos elementos, son utilizados con el fin de comprender la flexibilidad del aprendizaje automático [170].

Uno de los puntos más relevantes en el aprendizaje automático, es la flexibilidad, esto es debido, a que algoritmos de aprendizaje supervisado, pueden diferir en el espacio o dominio bajo el cual funcionan, provocando que un conjunto de datos se adapte a un algoritmo y no suceda lo mismo con otro. Debido al uso de diferentes algoritmos, meta-learning permite mejorar la medida de desempeño y aumentar la robustez del modelo. No obstante, la selección de los algoritmos puede influenciar negativamente en el desempeño [88].

Al utilizar diferentes tipos de metadatos, es posible aprender, seleccionar, alterar o combinar diferentes algoritmos de aprendizaje para resolver efectivamente un problema determinado [157].

Existen diferentes enfoques asociados al Meta-Learning y distintas formas de uso, los ejemplos más conocidos se basan en el uso de redes neuronales recurrentes [8], el aprendizaje reforzado [158] y los métodos de ensamble. En los diferentes casos planteados, se construyen modelos y se mejoran con respecto a elementos previos, generando diferentes iteraciones del proceso, aumentando la medida de desempeño y adquiriendo características que previamente no se consideraban [64].

2.1.10 Medidas de desempeño

Medir el desempeño del modelo predictivo es importante a la hora de evaluar qué tan efectivo es el entrenamiento o la clasificación que se genera, existen medidas que sólo se basan en la

cantidad de aciertos o errores que comete el clasificador, otras que implican la eficiencia del modelo y otras que se basan en la precisión.

A continuación, se define brevemente algunas de las medidas más utilizadas a la hora de evaluar modelos de aprendizaje supervisados:

- **Tasa de Verdaderos Positivos**: corresponde a la medida asociada a las correctas clasificaciones versus el total de clasificaciones realizadas, es decir, cuántas predicciones efectivas se obtuvieron con respecto a una clase.

- **Tasa de Falsos Positivos**: corresponde a la medida asociada a las clasificaciones mal efectuadas, es decir, cuántas predicciones erradas existen con respecto a una clase.

- **Accuracy**: corresponde al total de predicciones correctas con respecto al total de la muestra. Sea $\hat{y}_i$ el valor de predicción del ejemplo i e y_i corresponde al verdadero valor, la Accuracy se define como: $\texttt{accuracy}(y,\hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$

- **Precision**: es la capacidad del clasificador asociada a no etiquetar como positiva una muestra que es negativa, se define como: precision $= \frac{tp}{tp+fp}$, donde tp corresponde a verdaderos positivos y fp a los falsos positivos.

- **Recall**: es la capacidad del clasificador asociada encontrar todas las muestras positivas, se define como: recall $= \frac{tp}{tp+fn}$, donde tp corresponde a verdaderos positivos y fp a los falsos positivos.

- **F-β**: representa una ponderación armónica entre la Precision y el Recall, se define como: $F_\beta = (1+\beta^2)\frac{\text{precision}\times\text{recall}}{\beta^2\text{precision}+\text{recall}}$. , donde β un factor de ponderación.

- **Coeficiente de correlación de Matthews**: Se asocia a una medida de la calidad de las clasificaciones, la cual no se ve afectada por el desbalance de clases que pudiese existir, se define como $MCC = \frac{tp\times tn - fp\times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$, donde tp corresponde a verdaderos positivos y fp a los falsos positivos.

Las mediciones expuestas previamente, se utilizan para medir el desempeño de modelos de clasificación, mientras que para evaluar un estimador basado en respuestas continuas, normalmente se utilizan las siguientes:

- **Coeficiente de Pearson**: Medida lineal entre dos variables cuantitativas aleatorias que permite evaluar el grado de relación entre ellas, se encuentra en rangos entre -1 y 1 donde -1 indica que las variables no presentan relación y 1 que las muestras están estrechamente relacionadas. Se obtiene a partir de $\rho X,Y = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$

donde x_i representa los valores de predicción e y_i representa los valores reales de la muestra para n ejemplos.

- **Coeficiente de Spearman**: Medida de correlación que permite evaluar la asociación o relación entre dos muestras, su interpretación es similar al coeficiente de Pearson y se estima a partir de $\rho = 1 - \frac{6\sum D^2}{N(N^2-1)}$ donde D es la diferencia $x - y$ para el $i-th$ ejemplo y N es el total de ejemplos en la muestra.

- **Kendall τ rank**: Medida que permite evaluar la relación entre dos variables, su interpretación es similar a las basadas en coeficiente de Pearson y Spearman. Se obtiene a partir de $\tau = \frac{(\text{numbers of concordant pairs}) - (\text{number of discordant pairs})}{n(n-1)/2}$

- **Coeficiente de determinación R^2** *score*: es una medida que cuantifica cómo el predictor se adapta a nuevos ejemplos, posee un rango entre -1 y 1 donde -1 es lo peor y 1 lo mejor, esto es debido a que el estimador puede bajar su rendimiento. Se estima en base a $R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i-\hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i-\bar{y})^2}$ donde $\hat{y}_i$ corresponde al valor predicho para el $i-th$ ejemplo e y_i corresponde al valor real en una muestra de $n-samples$.

- **Error medio absoluto**: Estima la diferencia positiva entre el valor real y el valor predicho para un conjunto de ejemplos. Se estima a partir de $\text{MAE}(y,\hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1}|y_i - \hat{y}_i|$. donde $\hat{y}_i$ corresponde al valor predicho para el $i-th$ ejemplo e y_i corresponde al valor real en una muestra de $n-samples$.

- **Error cuadrático medio**: Estima la diferencia cuadrática entre el valor real y el valor predicho para un conjunto de ejemplos. Se obtiene a partir de $\text{MSE}(y,\hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1}(y_i-\hat{y}_i)^2$ donde $\hat{y}_i$ corresponde al valor predicho para el $i-th$ ejemplo e y_i corresponde al valor real en una muestra de $n-samples$.

- **Error logarítmico cuadrático medio**: Es similar al error cuadrático medio, la diferencia principal es que se utiliza el logaritmo natural de las diferencias entre respuesta y valor predicho. Se estima en base a $\text{MSLE}(y,\hat{y}) = \frac{1}{n_{\text{samples}}}\sum_{i=0}^{n_{\text{samples}}-1}(\log_e(1+y_i) - \log_e(1+\hat{y}_i))^2$ donde $\hat{y}_i$ corresponde al valor predicho para el $i-th$ ejemplo e y_i corresponde al valor real en una muestra de $n-samples$.

- **Error mediano absoluto**: es una medida robusta ante outliers, la pérdida o el error se calcula a partir de las medianas de las diferencias absolutas entre la respuesta y el valor predicho. Se estima en base a $\text{MedAE}(y,\hat{y}) = \text{median}(|\, y_1 - \hat{y}_1 \,|, \ldots, |\, y_n - \hat{y}_n \,|)$ donde $\hat{y}_i$ corresponde al valor predicho para el $i-th$ ejemplo e y_i corresponde al valor real en una muestra de $n-samples$.

2.1.11 Problemas asociados a los modelos de aprendizaje supervisado

Dentro de los principales problemas que pueden presentar los modelos de aprendizaje supervisado, se encuentran las situaciones en las que la cantidad de atributos que puede contener un set de datos es mucho mayor con respecto a la cantidad de ejemplos que se posee, conocido también como *"Maldición de la dimensionalidad"* [93].

Es posible definirla como: dado un conjunto de datos *nxm* con *n* ejemplos y *m* descriptores, si $m >> n$, es decir, si $m = n\ x\ n$, es posible que ocurra dicha problemática.

Posibles soluciones a este problema comprenden técnicas asociadas a reducción de dimensionalidad [156, 172], siendo las más utilizadas [3]:

- Métodos lineales de reducción como Análisis de componentes principales (PCA) y derivados.
- Análisis de características basados en modelos de clasificación/regresión aplicando Random Forest.
- Técnicas probabilísticas asociadas al Mutual Information.
- Evaluación de características relacionadas mediante coeficientes de Pearson o matrices de Correlación

En forma similar, también es posible que un set de datos contemple una gran cantidad de ejemplos y sus descriptores sean escasos. Estos casos se tratan con técnicas de reducción de dimensionalidad y contemplan la eliminación de ejemplos redundantes con el fin de maximizar la variabilidad de los ejemplos, técnicas como Mutual Information, Análisis de Correlaciones son bastante utilizadas en este problema.

Otro posible problema que se puede denotar es el sobreajuste [85], esto quiere decir, que el modelo es extremadamente complejo, por lo que éste se ajusta muy bien al set de entrenamiento. No obstante, a la hora de probar con nuevos set de datos no representa la performance obtenida.

Finalmente, un problema adicional a los modelos de clasificación se basa en el desbalance de clases [95]. Esto quiere decir, que existe una diferencia significativa entre los contadores de categorías asociadas a las clases, lo cual afecta a los algoritmos a la hora de entrenar, debido a que aumenta el riesgo de cometer errores del tipo I (falsos positivos). Normalmente, esto conlleva a una reducción de ejemplos de la clase mayoritaria en la etapa de entrenamiento o si es posible, la adición de nuevos ejemplos de la clase minoritaria.

[3] Estas técnicas, se explican en el capítulo 3

2.1.12 Validación de modelos

La validación de los modelos trata los problemas de sobreajuste y la generalización, es decir, evitar desarrollar modelos que sólo tengan buenas métricas o medidas de desempeño para los datos de entrenamiento y no permitan clasificar nuevos ejemplos.

Con el fin de poder evitar esta problemática, normalmente los set de datos se dividen en 3 conjuntos: Entrenamiento, validación y testeo. Esto quiere decir, se considera una porción de elementos para entrenar el modelo, una segunda instancia para obtener las medidas de desempeño y una tercera con el fin de determinar si el clasificador entrega resultados acorde a las respuestas conocidas [103].

Existen técnicas que a partir del set de entrenamiento, generan múltiples divisiones, con el fin de entrenar subconjuntos de elementos del conjunto de entrenamiento y así obtener modelos ponderados.

La subdivisión en n modelos para entrenar y generar modelos ponderados, permite aumentar la generalización del modelo, debido a que las medidas de desempeño varían levemente de aplicación en aplicación y al considerar las divisiones, se tienen distribuciones de las medidas del modelo. Dada las distribuciones, el entrenamiento reporta, la media de dicha distribución [74].

Dentro de las principales técnicas de validación se encuentra la Validación cruzada con k divisiones [74] y un caso particular conocido como *Leave one out*, en el cual el valor de k es igual a la cantidad de ejemplos en el entrenamiento [174]. Éstas se explican a continuación.

Validación Cruzada

La validación cruzada, a veces llamada estimación de la rotación, es una técnica de validación del modelo para evaluar cómo los resultados de un análisis estadístico se generalizarán a un conjunto de datos independiente.

Se utiliza principalmente en entornos donde la meta es la predicción, y se quiere estimar la precisión con la que un modelo predictivo se llevará a cabo en la práctica [74].

En un problema de predicción, a un modelo se le suele asignar un conjunto de datos, con respuestas conocidas, sobre los que se ejecuta el entrenamiento (conjunto de datos de formación) y un conjunto de datos desconocidos contra los que se prueba el modelo. El objetivo de la validación cruzada es definir un conjunto de datos para *probar* el modelo en la fase de entrenamiento (es decir, el conjunto de datos de validación), con el fin de limitar problemas asociados al sobre ajuste.

La idea es dividir el set de datos en K sub conjuntos, donde por cada subdivisión se entrena con elementos de tamaño $k-1$ y el conjunto restante, es usado para validar el modelo. Una explicación visual del problema, se puede explicar en la Figura 2.6:

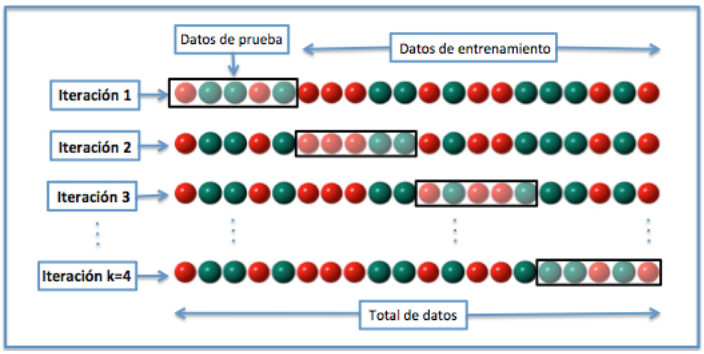


Fig. 2.6 Esquema representativo de validación cruzada.

A modo de ejemplo, sea $K = 10$ el número de divisiones a realizar, esto implica, que para un conjunto de datos de tamaño n se forman 10 sub conjuntos de tamaño $n/10$. La validación cruzada implica el entrenamiento y testeo del modelo K veces, en este caso, 10. Por cada iteración, se forma un conjunto de entrenamiento de tamaño $9n/10$ y se testea con un sub conjunto de tamaño $n/10$.

Por cada iteración, se rota el conjunto de entrenamiento y de testeo, y se obtiene una distribución del desempeño del modelo, reportando el promedio de las métricas asociadas.

Leave one out

Es un tipo especial de validación cruzada, en donde se tiene una muestra con n ejemplos en la etapa de entrenamiento se subdivide dicho set de datos considerando $n-1$ elementos, de tal manera que 1 no se considera, la idea en particular radica en entrenar con los $n-1$ ejemplos y validar o testear con el ejemplo restante, esto se itera n veces, tal como se expone en 2.7, implicando una mayor cantidad de iteraciones que validación cruzada, provocando además un mayor coste computacional [103].

Es decir, es un caso particular de validación cruzada, donde $K = n$ siendo n el número de ejemplos en el conjunto de datos.

Este tipo de validación, provoca disminuciones en las medidas de desempeño, ya que, aumentan la cantidad de puntos en la distribución. Además, en términos computacionales, resulta ser más costosa, por lo que se suele emplear con conjuntos de datos de tamaño bajo, normalmente, menores a 20 ejemplos.

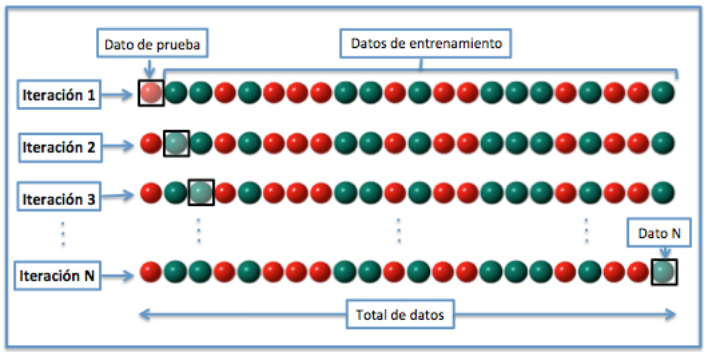


Fig. 2.7 Esquema representativo de Leave One.

2.2 Herramientas computacionales asociadas a evaluación de mutaciones

Las herramientas computacionales asociadas a la evaluación de mutaciones puntuales se centran principalmente en el análisis de cómo ésta afecta a la estabilidad o la predicción de energía libre asociada a los residuos involucrados en la mutación. Sin embargo, a pesar de que el objetivo es el mismo, se enfocan en diferentes puntos de vista para abordar la problemática, tanto a nivel de entrenamiento de modelos, cómo manipulación de set de datos, así como las técnicas utilizadas para la predicción de los cambios de energía libre.

En la Tabla 2.7 se exponen las principales herramientas existentes para la evaluación de la estabilidad de proteínas evaluando mutaciones puntuales, presentando las características, tipos de datos de entrada, resultados, estado de la herramienta y cuáles son las limitantes asociadas

Herr.	Características	Entradas	Salidas	Disp.
Foldx	Predice el valor del DDG a través del uso de funciones de energía derivados de términos fisicoquímicos, estadísticos e información estructural	Estructura en formato PDB e información sobre la mutación	Estimación de la diferenciade energía libre	Disponible mediante licencia académica

I-Mutant	Método basado en SVM para la predicción de DDG y la clasificación de la estabilidad de una proteína ante mutaciones puntuales. La mutación es caracterizada a través de propiedades estructurales y la información del ambiente. Permite la manipulación tanto de secuencias lineales como estructuras PDB	Secuencia lineal proteína, posición y mutación, en caso de existir estructura 3D, se requiere el archivo PDB	Predicción del DDG asociado a la mutación o clasificación de la mutación en estable o desestabilizante	Disponible para ejecución local
CUPSAT	Predice el DDG usando información estructural y del ambiente asociado a la mutación, además utiliza diferentes propiedades estructurales para estimar el valor de energía libre. Este es un método sólo basado en estimaciones utilizando técnicas de bioinformática estructural.	Estructura en formato PDB y la posición del residuo a mutar	Información sobre las 19 posibles sustituciones a realizar, referidas a términos como: ángulos de torsión, accesibilidad al solvente, tipo de estructura secundaria, dentro de las principales.	No disponible
Dmutant	Se basa en el uso de potenciales energéticos para entrenar modelos, utiliza distancias para describir el ambiente y estima el DDG asociado a la mutación	Estructura en formato PDB	Predicción del DDG asociado a la mutación	No disponible

AUTO-MUTE	Manipula las coordenadas de los residuos y aplica triangulación de Delaunay para forma geometrías, así permite describir el ambiente bajo el cual se encuentra el residuo. Los clasificadores se construyen entrenando con Random Forest y los predictores a través de árbolesde decisión.	ID-PDB, cadena y mutación	Predicción DDG o clasificación estabilidad, además de información relacionada al sector donde ocurre la mutación	Descargable para ejecución local

Table 2.7 Principales herramientas computacionales enfocadas a la evaluación de la estabilidad o predicción de cambios en la energía libre, asociado a mutaciones puntuales en proteína.

Diferentes son los puntos de vista que pueden ser considerados a la hora de evaluar el efecto que provoca la sustitución de residuos en la estabilidad de una proteína. Ya sea por medio de la estimación utilizando funciones de energía o potenciales energéticos (FoldX [161], Dmutant [188], CUPSAT [138]). Por otro lado, se encuentran métodos basados en algoritmos de apredizaje supervisado. No obstante, estos pueden ser diferenciados con respecto al algoritmo de entrenamiento utilizado o a la forma de describir la mutación. Por ejemplo, I-Mutant [37], utiliza SVM para predecir o clasificar el efecto de la mutación. Mientras que, AUTO-MUTE [124] utiliza Random Forest para la clasificación del efecto y Árboles de decisión como algoritmo de regresión.

Cada uno entrega distintas ventajas y desventajas, las cuales se basan principalmente, en la precisión del método y en el tiempo de cómputo relacionado al procedimiento. No obstante, también influye la disponibilidad de estos y si permiten descargas de código fuente para ejecuciones locales o simplemente disponen de versiones web.

2.2.1 Herramientas necesarias para la caracterización de los set de datos

Adicional a las herramientas expuestas, se hace una descripción breve de SDM [137] y MOSST [134], las cuales serán utilizadas a lo largo de la metodología con el fin de poder

caracterizar las mutaciones desde los puntos de vista termodinámico, aplicando SDM y filogenético, por medio de MOSST.

SDM

Site Directed Mutator (SDM) [137], es una de las herramientas más utilizadas a la hora de evaluar mutaciones puntuales en una proteína de interés y cuyo objetivo principal, es estudiar el efecto sobre la estabiliadd de la proteína que provoca el cambio del residuo.

Se basa en potenciales estadísticos de funciones de energía, para obtener una cuantificación del efecto de una sustitución de un residuo. Este valor se representa por la diferencia de energía libre de Gibbs ($\Delta\Delta G$). Para estimar el efecto, utiliza ambientes específicos de frecuencias de sustituciones de aminoácidos, sin la utilización de familias de proteínas homólogas.

Para realizar la estimación, SDM recibe un archivo PDB en el cual se describe la estructura de la proteína inicial, seguido además de la mutación, la cual se describe mediante *"W-Pos-M"*, los cuales corresponden a: Wilde residue, posición en la que se encuentra y mutate residue, respectivamente.

Sus medidas de desempeño en cuanto a la correlación entre los elementos predichos y los reales alcanza un 0.8, calculada a partir del testeo de la herramienta con mutaciones en la proteína Barnasa y staphylococcal nuclease. Este desempeño, lo convierte en una herramienta bastante precisa a la hora de estudiar nuevas predicciones.

Una de las principales razones de utilizar SDM y no otras herramientas computacionales, como las expuestas en la Tabla 2.7, es el hecho de basarse en potenciales estadísticos. Si bien, esta metodología no es tan precisa como el uso de potenciales físicos de funciones de energía, el hecho de utilizar simulaciones basadas en mecanismos de Monte Carlo, conlleva un gran costo computacional en horas de cálculo y recursos. No obstante, es mucho más eficiente que los métodos basados en Machine Learning, dado a que estos, normalmente, tienden a ajustarse y se requiere una gran cantidad de datos para entrenar estos modelos. Además, ya que estos se basan principalmente en métodos asociados a Support Vector Machine (SVM) o Redes Neuronales. Los primeros, no son capaces de adaptarse a espacios altamente no lineales. Mientras que los basados en redes neuronales, no permiten aprender de los atributos empleados, debido a la forma en cómo estos trabajan.

Además, SDM entrega resultados adicionales que permiten comprender el ambiente bajo el cual se produce la mutación y cómo, propiedades termodinámicas se ven afectadas ante la sustitución de residuos.

MOSST

Mutagenesis Objetive Search and Selection Tool (MOSST) [134], es una herramienta que permite analizar una proteína de interés, con respecto a un conjunto de proteínas con relación filogenética, representadas en un alineamiento múltiple de secuencias. Esto con el fin, de poder detectar posiciones en la proteína de interés o target, que podrían ser mutadas para alterar o no las características de la misma.

Por otro lado , permite estimar mutagénesis relacionadas con la posibilidad de si un cambio de aminoácido dado tendría efectos perjudiciales sobre la proteína.

Además, como un uso alternativo, es factible la identificación de nsSNPs cuyos fenotipos son relevantes en una familia de genes, permitiendo separar, a aquellas sustituciones que no tienen implicaciones a nivel de funcionalidad.

MOSST aplica técnicas estadísticas para el análisis de las posiciones y se centran en comprender los efectos de las sustituciones desde el punto de vista filogenético. Esto es relevante, ya que como input, sólo necesita secuencias lineales de proteínas, es decir, no es necesario el uso de estructuras en formato PDB, lo cual permite abarcar un mayor número de proteínas de estudio.

Actualmente MOSST se encuentra disponible vía ejecución local, implementado bajo lenguaje de programación Matlab. No obstante, posee una versión online para su uso, facilitando el acceso a la herramienta a diferentes estratos de público.

Utilizar la herramienta MOSST como generación de descriptores basados en propiedades filogenéticas, radica en las ventajas que ésta presenta. Por un lado, permite evaluar la propensión de la mutación y cómo ésta afecta a las características de la proteína en estudio y su familia. Además, permite entender conceptos relacionados a mutaciones, que no son considerados al utilizar propiedades termodinámicas y estructurales. Ya que, estos últimos, sólo evalúan el ambiente bajo el cual ocurre la mutación, mientras que MOSST, evalúa la propensión al cambio.

El uso de las herramientas MOSST y SDM permitirán describir las mutaciones desde los puntos de vista filogenético y termodinámico-estructural, de tal manera, las falencias de cada método, se complementan, facilitando la generación de descriptores relevantes para las mutaciones.

2.3 Hipótesis

En base a las herramientas existentes y en vista del aumento considerable de datos asociados a mutaciones en proteínas y el conocimiento de las respuestas que éstas generan, se evidencia la necesidad del desarrollo de herramientas computacionales o nuevos modelos de clasificación

o regresión que faciliten el entrenamiento de proteínas singulares y la evaluación de sus mutaciones puntuales, con el fin de poder evaluar nuevos ejemplos y cuáles serían los efectos de estos, sin tener que recurrir en grandes costos económicos y tiempos de espera.

Dado esto se propone la siguiente hipótesis.

El uso de propiedades filogenéticas, termodinámicas y estructurales como descriptores de mutaciones permite el desarrollo de modelos predictivos inspirados en sistemas de meta-learning?

Además de la hipótesis central surgen interrogantes como.

- Es posible utilizar estos nuevos modelos como herramientas para diagnóstico médico?
- Cómo se evalúan la robustez y la generalización de estos modelos, serán capaces de adaptarse a nuevos ejemplos?
- Es factible el desarrollo de una herramienta computacional que permita entrenar diferentes set de datos y que facilite la predicción de nuevos ejemplos?

2.4 Objetivos

En base a la hipótesis planteada y a las preguntas adicionales expuestas, se exponen a continuación el objetivo general y los objetivos específicos.

2.4.1 Objetivo general

Diseñar e implementar estrategias inspiradas en Meta Learning para la implementación de modelos de clasificación y regresión, asociados a mutaciones puntuales en proteínas de interés basados en descriptores termodinámicos, estructurales y filogenéticos.

2.4.2 Objetivos específicos

Dentro de los objetivos específicos se encuentran los siguientes.

1. Preparar y describir, por medio de propiedades termodinámicas, estructurales y filogenéticas, set de datos de mutaciones puntuales de proteínas con respuesta conocida expuestos en bibliografía o bases de datos públicas.

2. Implementar y evaluar metodología de meta learning para el diseño de meta modelos de clasificación y regresión de mutaciones puntuales aplicados a set de datos de proteínas generadas.

3. Diseñar e implementar herramienta computacional que permita el entrenamiento de set de datos y el uso de meta modelos para la evaluación de nuevos ejemplos.

4. Testear y evaluar comportamiento de la herramienta y los meta modelos en base a sistemas de datos que involucren mutaciones en proteínas con respuesta conocida.

5. Implementar modelos de clasificación para la relevancia clínica de mutaciones puntuales en proteína pVHL, asociada a la enfermedad von Hippel Lindau.

2.5 Metodología propuesta

Con el fin de poder responder a la hipótesis planteada y dar solución a los objetivos impuestos, se propone una metodología general, en la cual, se consideran diferentes estrategias, implementaciones y evaluación de modelos. A continuación se explica la metodología propuesta y los componentes principales de ésta.

2.5.1 Preparación de set de datos

La preparación del set de datos consiste en obtener data para poder entrenar los modelos predictivos, la data se asocia a información de mutaciones en proteínas y la respuesta que ésta genera. En la Figura 2.8 se expone un esquema general con los pasos desarrollados para la preparación del set de datos.

Tal como se expone en la Figura 2.8, los set de datos se buscan en la bibliografía, a partir de modelos desarrollados previamente, bases de datos, en la literatura, etc. El objetivo fundamental, es encontrar proteínas con mutaciones puntuales cuyo efecto sea conocido, dicha respuesta puede ser categórica, es decir, asociada al diseño e implementación de modelos de clasificación o continua y se aplica para modelos de regresión.

En una segunda instancia, a partir de la data recolectada, ésta se procesa con el fin de poder obtener set de datos de proteínas individuales con una cantidad de ejemplos considerables que permitan el diseño de modelos válidos, para ello, fueron implementados scripts bajo el lenguaje de programación Python con el fin de recuperar las proteínas, obtener la información y generar la data de manera individual, además, eliminar ejemplos ambiguos. Es decir, filas con los mismos valores pero cuya columna de respuesta fuese diferente.

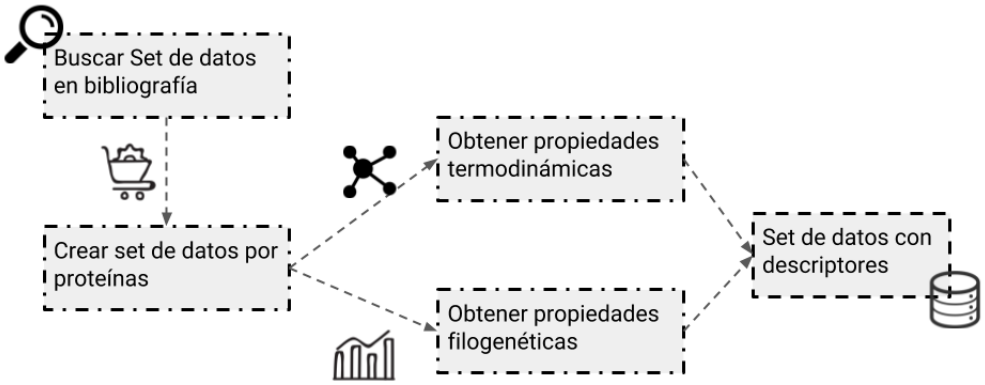


Fig. 2.8 Esquema representativo asociado al proceso de generación de set de datos de mutaciones puntuales en proteínas.

A partir de esto, se forman n set de datos asociados a n proteínas, cada uno con m ejemplos y cuyos descriptores consisten en el residuo original, posición en proteína, residuo mutado y la respuesta asociada. El desbalance de clases se analiza con respecto a las posibles categorías existentes en la respuesta y el porcentaje de representatividad que éstas poseen en la muestra. Se considera que el set de datos exhibe este comportamiento cuando presentan las características expuestas en la sección 2.1.11. En el caso de detectar esta problemática, el conjunto de datos será tratado empleando el método SMOTE (Synthetic Minority Oversampling Technique) [44].

Posteriormente, se aplican las herramientas SDM [137] y MOSST [134] con el fin de obtener los descriptores asociados a las propiedades termodinámicas y filogenéticas, respectivamente. Para ello, scripts implementados en lenguaje de programación Python, son desarrollados para consumir los servicios de dichas herramientas y registrar los resultados obtenidos, formando así, set de datos con los descriptores planteados en los objetivos iniciales. Un punto importante a destacar, es que el uso de SDM implica que las proteínas a trabajar, deben presentar una estructura 3D reportada en el Protein Data Bank [18] o al menos poseer un modelo representativo y validado. Esto es debido a que se utilizan informaciones de coordenadas para la estimación del efecto de la mutación, minimizaciones energéticas y estabilización de la mutante.

Ya con los descriptores formados, las características asociadas a variables categóricas son codificadas. Si la totalidad de posibles categorías supera el 20% del total de características en

el set de datos, se aplica Ordinal Encoder, en caso contrario, One Hot Encoder [140]. Ordinal Encoder consiste en la transformación de variables categóricas en arreglos de números enteros con valores desde $0, ..., n-1$ para n posibles categorías. Por otro lado, One Hot Encoder, consiste en agregar tantas columnas como posibles categorías existan en el set de datos completadas mediante binarización de elementos (0 si la característica no se presencia, 1 en caso contrario)[4].

Es importante mencionar, que las respuestas asociadas a las mutaciones pueden ser del tipo continuo o categórico, lo cual implica que tanto los modelos como las métricas varían. No obstante, se aplica la metodología indistintamente, con el fin de demostrar la robustez del método y la eficacia de éste sin importar el tipo de modelo que se éste entrenando.

2.5.2 Implementación de meta modelos de clasificación/regresión

La implementación de meta modelos consiste en la obtención de un grupo de estimadores que en conjunto, permiten clasificar o predecir nuevos ejemplos. Para ello, se diseña e implementa una metodología inspirada en Sistemas de Meta Learning y aplicando técnicas estadísticas para la evaluación del desempeño y el uso del meta modelo con nuevos ejemplos.

En la Figura 2.9, se exponen las etapas asociadas a la implementación de meta modelos, contemplando desde la fase de entrenamiento de los modelos hasta la unión en meta clasificadores, lo cual se reporta en la herramienta MLSTools (Paper en redacción).

Cada una de las etapas, contempla un conjunto de scripts implementados en lenguaje de programación Python y empleando la librería Scikit-Learn para el entrenamiento y evaluación de los clasificadores o predictores [140], así como Numpy para el uso de módulos estadísticos [178].

Tal como se observa en la Figura 2.9, es posible identificar etapas claves en el proceso: Exploración de modelos, Selección y Generación de los meta clasificadores/predictores, junto con su evaluación. Cada una de estas etapas se exponen a continuación.

Exploración de modelos

La exploración de modelos o estimadores, se basa en la aplicación de diferentes algoritmos de aprendizaje supervisado con variaciones en sus parámetros de configuración inicial. La utilización de los algoritmos, depende principalmente del tipo de respuesta que presente el set de datos, es decir, si es continua o categórica. No obstante, a modo resumen, en la Tabla

[4]Más detalles sobre la codificación de variables categóricas y secuencias lineales de proteínas serán tratadas en el capítulo 3.

Fig. 2.9 Esquema representativo asociado al proceso de creación de meta modelos utilizando la metodología reportada para la herramienta MLSTools (Paper en redacción).

2.8 se exponen los algoritmos utilizados, el caso en el que se usan y los parámetros que se varían junto con el total de iteraciones posibles para cada elemento:

Algoritmos y parámetros empleados en la etapa de Exploración en MLSTools					
#	**Algoritmo**	**Tipo**	**Parámetros**	**Uso**	**Iteraciones**
1.	Adaboost	Ensamble	Algoritmo Número estimadores	Clasificación y Regresión	16
2.	Bagging	Ensamble	Bootstrap Número estimadores	Clasificación y Regresión	16
3.	Bernoulli Naive Bayes	Probabilístico	Default	Clasificación	1
4.	Decision Tree	Características	Criterio división Función de impureza	Clasificación y Regresión	4
5.	Gaussian Naive Bayes	Ensamble	Default	Clasificación y Regresión	1
6.	Gradient Tree Boosting	Ensamble	Función de pérdida Número estimadores	Clasificación y Regresión	16
7.	k-Nearest Neighbors	Distancias	Número Vecinos Algoritmo Métrica distanciaPesos	Clasificación y Regresión	160

9.	Nu Support Vector Machine	Kernel	Kernel Nu Grado polinomio	Clasificación y Regresión	240
10.	Random Forest	Ensamble	Número estimadores Función de impureza Bootstrap	Clasificación y Regresión	32
11.	Support Vector Machine	Kernel	Kernel C Grado polinómio	Clasificación y Regresión	240
				Total Iteraciones	**726**

Table 2.8 Tabla resumen, algoritmos implementados, parámetros utilizados e iteraciones involucradas por cada algoritmo.

Como se observa en la Tabla 2.8, son sobre 720 modelos los que se generan y a partir de ellos se obtiene distribuciones de medidas de desempeño que permiten evaluarlos. En el caso de modelos de regresión se utilizan los coeficientes de Pearson, Spearman, Kendall τ y R^2, mientras que para modelos de clasificación, se consideran la Precisión, Exactitud, Recall y F1.

Finalmente, esta etapa, entrega set de modelos entrenados y evaluados según las métricas de interés. Se destaca que cada modelo es validado a través del proceso de validación cruzada, con el fin de poder disminuir posibles sobreajustes. El valor de k asociado a las subdivisiones a realizar varía con respecto a la cantidad de ejemplos que presente el set de datos, es decir, sea n la cantidad de ejemplos en la muestra, si $n <= 20$ se tiene que $k = n$ implicando el uso de Leave one out, si $n > 20$ y $n <= 50$ se considera un valor de $k = 3$, si $n > 50$ y $n <= 100$ $k = 5$, por último, si $n > 100$ se tiene un valor de $k = 10$.

Selección de modelos

Cada distribución de medida de desempeño perteneciente a los modelos entrenados en la fase de Exploración, se somete a test estadísticos basados en Z-score [140] que permite seleccionar los modelos cuyas métricas representen outliers positivos dentro de la distribución.

El algoritmo general, utilizado para el desarrollo de esta selección es como se expone en el algoritmo 1, para el cual se detallan los pasos simplificados que permiten obtener un conjunto de modelos entrenados y que representan los valores más altos dentro de su distribución. Es importante mencionar, que se obtiene un conjunto M' con los modelos, considerando como punto de selecciones los valores evaluados con respecto a la desviación

estándar, considerando los umbrales 3 σ, 2 σ y 1.5 σ por sobre la media, si ningún factor se cumple, sólo se considera el valor máximo en la distribución.

Es importante mencionar, que cada distribución puede permitir la selección de distintos modelos, lo cual implica que un mismo modelo pueda ser seleccionado en diferentes medidas, razón por la cual, a la hora de obtener el conjunto de modelos M' se remueven aquellos elementos que se encuentran repetidos. Siendo estos, sólo los modelos que presenten igualdad tanto en el algoritmo como en sus parámetros de configuración inicial.

Algoritmo 1 Algoritmo de selección de modelos

Entrada: Conjunto M con modelos entrenados y sus medidas de desempeño, Lista L con medidas de desempeño.

Salida: Conjunto M' con modelos seleccionados.

para i en L **hacer**
  Calcular media μ, desviación estándar σ en distribución M_i
  para $x \in M_i$ **hacer**
    si $x \geq \mu + 3 * \sigma$ **entonces**
      Agregar x a M'
    fin si
  fin para
  si largo $M' = 0$ **entonces**
    para $x \in M_i$ **hacer**
      si $x \geq \mu + 2 * \sigma$ **entonces**
        Agregar x a M'
      fin si
    fin para
    si largo $M' = 0$ **entonces**
      para $x \in M_i$ **hacer**
        si $x \geq \mu + 1.5 * \sigma$ **entonces**
          Agregar x a M'
        fin si
      fin para
      si largo $M' = 0$ **entonces**
        para $x \in M_i$ **hacer**
          si $x = MAXM_i$ **entonces**
            Agregar x a M'
          fin si
        fin para
      fin si
    fin si
  fin si
fin para
devolver D sin valores extremos

Generación y evaluación de meta modelos

A partir del conjunto de modelos M', el cual representa los estimadores seleccionados, cuyas medidas de desempeño son las más altas en sus distribuciones correspondientes, se generan meta modelos, es decir, estimadores compuestos de diversas unidades, los cuales en conjunto entregan una respuesta, ya sea por ponderación o votación. El proceso general para la generación de los meta modelos, es descrito a continuación.

En una primera instancia, los modelos son nuevamente entrenados y se comparan las nuevas medidas de desempeño con las obtenidas previamente. En caso de que exista una diferencia mayor al 20%, en cualquiera de sus métricas, el modelo se remueve del conjunto M'. La razón fundamental de esto, es debido a que se espera desarrollar modelos robustos cuyas evaluaciones no presenten variaciones significativas y que realmente no alteren sus predicciones ante nuevos ejemplos, razón por la cual, se aplica nuevamente validación cruzada para validar los modelos.

Con el fin de evaluar el desempeño de los meta modelos, nuevas medidas se generan a partir de la información resultante de los modelos individuales. No obstante, la forma en la que se obtienen varían dependiendo del tipo de respuesta que se debe entregar.

Si la respuesta es continua, se obtiene los valores de predicción de cada modelo y se promedian, para luego aplicar las métricas estándar (Coeficiente de Pearson, Kendall τ, Spearman y R^2) sobre estos valores promediados y los reales. Expresado matemáticamente:

Sea M' la cantidad de elementos en el meta modelo, n la cantidad de ejemplos en el set de datos y sea Y el vector de respuestas reales de tamaño n. Para cada $M'_i \in M'$ se obtiene un vector Y_i que representa los valores de predicción entregados por el modelo M'_i. A partir de cada Y_i se genera una matriz de predicciones $P(mxn)$ donde m representa la cantidad de modelos en M'. Finalmente, se obtiene un vector Y' de tamaño n, el cual se compone de la media de cada columna en la matriz P, es decir, para el ejemplo i se obtienen m predicciones, las cuales son promediadas, formando el valor $Y'_i \in Y'$. Vector el cual, se utiliza para obtener las métricas de desempeño.

Para el caso en que la respuesta sea categórica, es decir, los modelos son del tipo clasificación, se obtiene la respuesta de cada modelo individual y se selecciona una única categoría, correspondiente a aquella que presente una mayor probabilidad de ocurrencia dada la distribución de elementos y considerando para ello las probabilidades iniciales de cada categoría en el set de datos de estudio. De esta forma, se obtiene un vector respuesta con la clasificación de cada ejemplo cuyo valor corresponde al evento más probable a ocurrir, este vector se compara con el set de respuestas reales y se aplican las métricas de interés para clasificadores.

2.5.3 Cómo usar los meta modelos para la clasificación de nuevos ejemplos?

Nuevos ejemplos pueden ser clasificados o predecir su respuesta, dependiendo sea el caso, a partir de los meta modelos desarrollados. En el caso de estimadores basados en variables continuas, los nuevos ejemplos se someten a cada uno de los modelos individuales pertenecientes al sistema, los cuales generan una respuesta individual, a partir de dichas respuestas, se genera un intervalo de confianza con un nivel de significancia $\alpha = 0.01$ donde existe una mayor probabilidad de que se encuentre el valor real de la predicción dado los valores del entrenamiento. Para ejemplos que impliquen clasificación, se obtiene la respuesta de cada modelo individual y se evalúa la probabilidad de ocurrencia de cada categoría, entregando así, la respuesta condicionada por una probabilidad de ocurrencia del evento.

2.5.4 Uso de meta modelos en sistemas de proteínas

El objetivo principal de esta metodología, radica en el hecho de crear una herramienta que permita implementar modelos basados en algoritmos de aprendizaje supervisado para set de datos de mutaciones puntuales o variantes para una misma proteína.

Un flujo general del uso de la herramienta, se expone en la Figura 2.10.

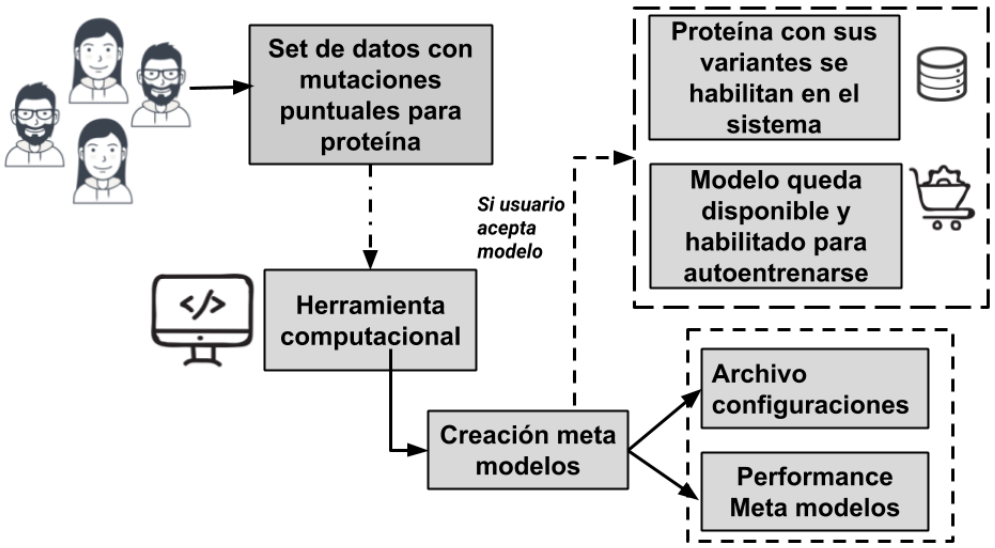


Fig. 2.10 Esquema representativo de flujo asociado a la herramienta de generación de meta modelos para mutaciones puntuales en proteínas de interés.

La idea general, consiste en que usuarios de la herramienta, puedan entrenar sus propios modelos de clasificación o regresión, basados en la metodología expuesta en los pasos

anteriores mediante el uso de Meta Learning Sytem Tools (Paper en Redacción). Para ello, los usuarios deben entregar sus set de datos con la información necesaria para ser procesada: cadena, residuo original, posición, residuo mutado y respuesta o efecto de la mutación, además del archivo PDB a ser procesado.

La herramienta, aplica los pasos expuestos en la metodología de este capítulo generando un meta modelo basado en algoritmos de aprendizaje supervisado y las medidas de desempeño que permiten evaluar el modelo obtenido.

Si el usuario acepta la metodología y permite la publicación de los datos, el sistema habilita el acceso tanto a los meta modelos como a los set de datos y los agrega a la lista de procesos de modelos auto entrenables.

Esto último, implica que ante la adición de nuevos ejemplos al set de datos, el sistema actualiza los modelos y las medidas de desempeño, aplicando la metodología expuesta, así, constantemente mantiene la actualización de la información y permite mantener en constante crecimiento los datos que contemplan el desarrollo de los modelos.

2.6 Análisis y evaluación de los set de datos a utilizar

A continuación, se exponen los resultados obtenidos hasta el momento, presentando principalmente, los set de datos a utilizar, las características que estos poseen y qué representan, con el fin de contextualizar la data a manipular y los modelos a generar.

2.6.1 Set de datos utilizados

En el presente apartado se describen las características básicas de los set de datos trabajados, así como también, qué representan las proteínas bajo las cuales se están desarrollando los modelos de estimadores.

Descripción general

Los set de datos utilizados, tanto para la formación de los inputs asociados al sistema, así como también la validación de respuesta correspondiente a la mutación que estos tienen, fueron extraídos desde distintas bases de datos de mutaciones en proteínas de estudios relacionados a los cambios que provoca la sustitución del residuo inicial, ya sea a nivel de cambios energéticos o estabilidad de la proteína.

11 set de datos con respuesta continua fueron obtenidos. Cada set de datos contemplaba como elemento a predecir, las diferencias de energía libre de Gibbs, entre los residuos

originales y mutados. Las mutaciones fueron seleccionadas desdes diversos estudios en los cuales se reportaron, centrándose en [177, 163, 143, 3, Bordner and Abagyan].

Adicional a los set de datos con respuesta continua, 8 conjuntos de elementos asociados a tareas de clasificación fueron obtenidos desde diversos estudios reportados a la actualidad [7, 31, 38, 147, 37, 145, 102, 123, 73].

De tal manera, se generó un total de 19 conjuntos de set de datos, con respuesta categórica y continua, los cuales se asocian a proteínas independientes, usadas para la evaluación de las metodologías planteadas. Estas 19 proteínas junto con su descripción, se exponen en la Tabla 2.9.

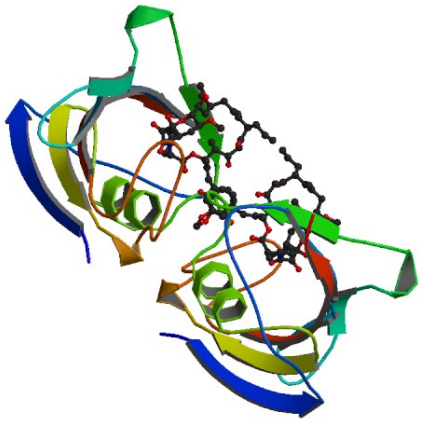
Resumen set de datos de proteínas y sus características				
#	**Código PDB**	**Tipo**	**Ejemplos**	**Descripción**
1.	1A22	Regresión	132	Human growth hormone bound to single receptor
2.	1CH0	Regresión	191	Crystal and molecular structures of the complex of alpha-*Chymotrypsin with its inhibitor Turkey Ovomucoid third domain
3.	1DKT	Regresión	119	CKSHS1: Human cyclin dependent kinase subunit, type 1 complex with metavanadate
4.	1FKJ	Regresión	219	Atomic structure of FKBP12-FK506, an immunophilin immunosupressant complex
5.	1FTG	Regresión	203	Structure of apoflavodoxin: closure of a Tyr/Trp aromatic gate leads to a compact fold
6.	1PPF	Regresión	190	X-Ray crystal structure of the complex of human leukocyte elastase and the third domain of the Turkey ovomucoid inhibitor
7.	1RX4	Regresión	556	Dihydrofolate reductase (E.C.1.5.1.3) complexed with 5,10-Dideazatetrahydrofolate and 2'-Monophosphadenosine 5'-Diphosphoribose
8.	1WQ5	Regresión	239	Crystal structure of tryptophan synthase alpha-subunit from Escherichia coli
9.	2AFG	Regresión	134	Human acidic fibroblast growth factor

10.	3SGB	Regresión	191	Structure of the complex of Streptomyces Griseus protease B and the Third domain of the Turkey ovomucoid inhibitor
11.	5AZU	Regresión	203	Crystal structure analysis of oxidize Pseudomonas Aeruginoa Azurin at PH 5.5 and PH 9.0. A PH-induced conformational Transition involves a peptide bond flip
12.	1BN1	Clasificación	1802	Carbonic anhydrase II inhibitor
13.	1BVC	Clasificación	561	Structure of a Biliverdin Apomyoglobin complex
14.	1LZ1	Clasificación	848	Human Lysozyme. Analysis of Non-Bonded and Hydrogen-Bond interactions
15.	1STN	Clasificación	2193	The crystal structure of Staphylococcal Nuclease
16.	1VQB	Clasificación	820	Gene V Protein (Single-Stranded DNA Binding Protein)
17.	2CI2	Clasificación	741	Crystal and molecular structure of the Serine proteinase inhibitor CI-2 from Barley seeds
18.	2LZM	Clasificación	2336	Structure of Baceriophage T4 Lysosyme
19.	2RN2	Clasificación	712	Structural details of ribonuclease H from Escherichia Coli

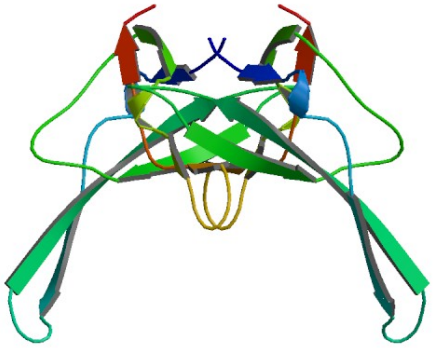
Table 2.9 Resumen de proteínas utilizadas para el desarrollo de meta modelos basados en metodología Meta Learning System propuesta durante este capítulo.

Cada una de las proteínas presentan diferentes características y funcionalidades, algunas facilitan la unión a DNA, mientras que otras presentan propiedades enzimáticas, por otro lado, existen enzimas que representan inhibidores, entre las principales. Esto es interesante a la hora de evaluar el poder que presenta la metodología con respecto al análisis de diferentes proteínas, estructuras y complejos, ya que se presenta una gran variedad en cuanto a forma y funcionalidad de éstas, lo que implica que el sistema no se limita por cierto tipo de estructuras o complejos.

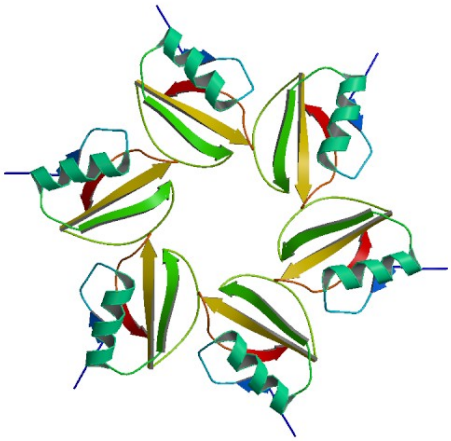
A modo de ilustrar las diferencias estructurales de las proteínas en estudio, en la Figura 2.11 se exponen algunas de las estructuras asociadas a las proteínas utilizadas para desarrollar modelos de clasificación o regresión.



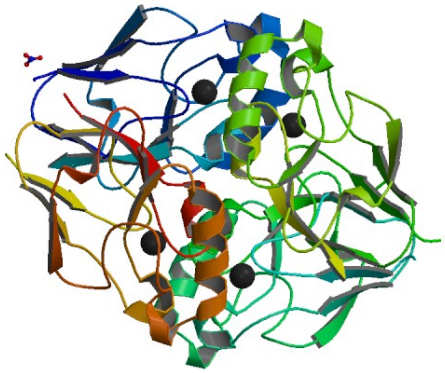
(a) 1FKJ



(b) 1VQB



(c) 2CI2



(d) 5AZU

Fig. 2.11 Representación de estructuras de proteínas ejemplos utilizadas para el desarrollo de meta modelos de clasificación.

Las mutaciones fueron recolectadas desde diferentes set de datos, por lo que, en caso de información ambigua, es decir, una misma mutación con diferentes respuestas, no fueron consideradas. Por otro lado, debido a que para la aplicación de la herramienta SDM se necesitaba la cadena a la cual pertenece en residuo, scripts desarrollados en Python y utilizando la librería BioPython, permitieron procesar los archivos asociados a las estructuras de las proteínas, identificadas desde el Protein Data Bank (PDB) [2]. Descartando aquellas mutaciones reportadas en las que no se encontró la cadena, obteniendo como resultante la cantidad de mutaciones reportadas para cada proteína expuestas en la Tabla 2.9.

Evaluación del desbalance de clases y distribución de respuestas continuas

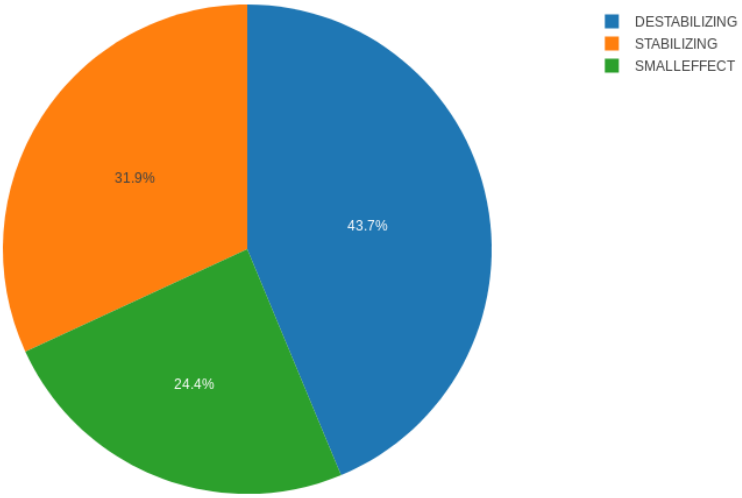
El desbalance de clases se evaluó en aquellos set de datos con respuesta categórica, lo cual contempla, tres posibles casos: Neutral, Estable, No Estable, esto es debido a que todos los estudios donde se evalúan mutaciones, normalmente se analizan cambios que alteren la estabilidad de la proteína. En la Figura 2.12, se aprecia a modo ejemplo dos set datos y su distribución de categorías para la variable respuesta.

En las 8 proteínas en estudio para modelos de clasificación, la distribución de las categorías es similar a lo expuesta en la Figura 2.12 para todas ellas, donde cerca del 50% corresponden a mutaciones que afectan positivamente a la estabilidad, mientras que mutaciones que provocan cambios negativos o no generan diferencias, se encuentran en proporciones similares, ambas cercanas al 25%.

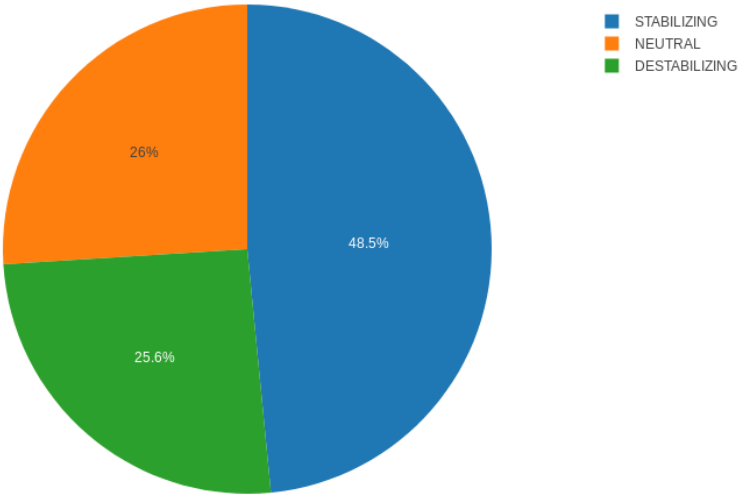
Si bien las proporciones son dispares, para este caso, se considera un desbalance como un elemento que representa menos de un 5% del total de la muestra, además, dado a que la cantidad de ejemplos son elevadas, un 20% o un 25% implica cerca de 200 mutaciones, en promedio, que cumplen dicha característica. También, el hecho de que exista una cantidad inferior de mutaciones no benéficas a la estabilidad viene dada a la dificultad de encontrar y reportar mutaciones que afecten negativamente a para una proteína, es debido a la propensión filogenética [134] que estos ocurran, lo cual se ve reflejado en las diferencias asociadas a cambios positivos dentro del set de mutaciones. No obstante, si bien el hecho de que la propensión filogenética indique que el cambio tiende a mejorar estabilidad, diseñar mutantes con mejoras en propiedades de interés, es un problema latente en la actualidad, de alto costo económico y computacional y con una gran demanda desde diferentes áreas del conocimiento.

En los set de datos para el desarrollo de modelos de regresión, se evaluó la distribución de la respuesta, en este caso, valores de $\Delta\Delta\, G$ asociado a diferencias de energía libre producidas entre el residuo mutado y el original, tal que: $\Delta\, Res_{mut} - \Delta\, Res_{wild} = \Delta\Delta\, G$.

Las distribuciones se evaluaron utilizando el test de Shapiro, con el fin de determinar si la distribución se comportaba como una normal. Para todas las proteínas estudiadas, en



(a) 1STN

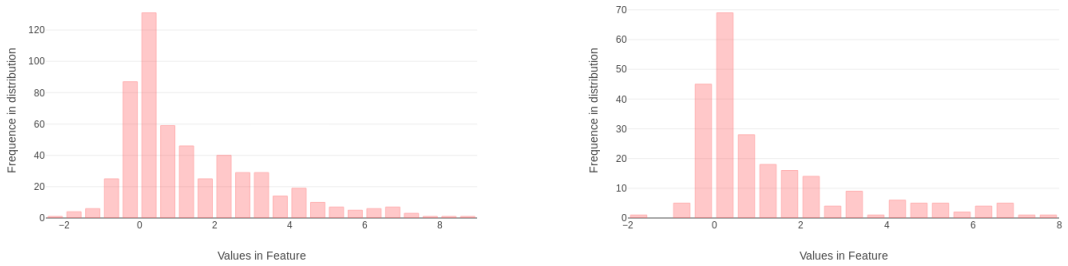


(b) 2RN2

Fig. 2.12 Evaluación del desbalance de clases en proteínas ejemplo.

los 11 set de datos, las respuestas presentaron distribución normal, con valores de Shapiro sobre 0.8 y un p-value ≤ 0.01, lo cual indica una alta confianza estadística en los resultados presentados por dicho test.

Una visualización de las distribuciones puede generarse a partir del desarrollo de histogramas, los cuales, a modo de ejemplo se expone en la Figura 2.13.



(a) Histograma para respuesta continua en 1RX4 (b) Histograma para respuesta continua en 1WQ5

Fig. 2.13 Evaluación de la distribución de respuesta continua en set de datos de proteínas.

El análisis de estas características es relevante a la hora de diseñar modelos de clasificación o regresión, debido a que si existe una tendencia por una clase condiciona al clasificador a *"aprender en base a la mayoría"*, por lo que puede aumentar los errores en cuanto a falsos positivos, dado a que, no se tiene la cantidad de ejemplos suficientes para una clase que permitan al modelo capturar las posibles variaciones asociadas a ésta.

Dado a los análisis de evaluación de representatividad de categorías en el set de datos y distribución de respuestas continuas, se expone que los set de datos seleccionados no presentan desbalance significativo para el caso de desarrollo de modelos de clasificación y a su vez, todas las respuestas asociadas a cambios en la energía libre para modelos de regresión, presentan distribución normal. Razón por la cual, es factible el desarrollo de modelos asociados a las respuestas presenten en los set de datos seleccionados. No obstante, sólo se ha considerado el problema del desbalance y la evaluación de distribución en la respuesta continua, una vez caracterizado los set de datos a partir de las propiedades fisicoquímicas y termodinámicas, se analizarán las características y cómo éstas condicionan la clasificación o la predicción de cambios energéticos.

Chapter 3

Digitalización de secuencias lineales de proteínas aplicadas al reconocimiento de patrones y modelos predictivos

Desarrollar modelos predictivos basados en algoritmos de aprendizaje supervisado, o, la identificación de patrones aplicando técnicas de clustering, son tareas muy relevantes a la hora de trabajar con secuencias de proteínas, ya sea para identificar grupos con características comunes o entrenar modelos predictivos de respuestas de interés. En ambos casos, se requiere el uso de conjuntos de datos altamente informativos y con características numéricas para poder utilizar los métodos implementados en las librerías actuales [140].

Diferentes metodologías se han implementado, para manipular las variables categóricas en set de datos y lograr su codificación numérica. Enfoques basados en adición de columnas según las categorías o simple transformación empleando representaciones en conjuntos naturales, suelen ser utilizados. No obstante, generan bastante discusión sobre las nuevas representaciones. y a su vez, el hecho de aumentar el número de columnas, conlleva a incrementar las dimensiones del conjunto de datos, provocando efectos en los desempeños de los algoritmos [140].

Particularmente, en secuencias de proteínas, se han utilizado las frecuencias de incidencia de los residuos para codificarlos, la cual, pese a su simplicidad, ha resultado ser efectiva en diferentes casos de uso [136]. No obstante, este tipo de codificación, no permite explorar el ambiente bajo el cual se encuentran los residuos y tampoco considera el efecto de propiedades fisicoquímicas ni termodinámicas.

En diferentes estudios, los residuos se describen a partir de sus propiedades fisicoquímicas y adicional a ello, se emplea información que permite describir el ambiente del residuo a caracterizar, empleando binarizaciones para describir los residuos cercanos, ya sea por medio

del uso de un rango espacial, utilizando modelos o estructuras tridimensionales en donde se representan las coordenadas espaciales de los residuos, o empleando un rango lineal en secuencias lineales de proteínas [36, 38].

Un enfoque basado en las propiedades fisicoquímicas en combinación con la aplicación de transformaciones de Fourier, ha permitido demostrar que ciertos residuos permiten entregar las características asociadas a la propiedad en estudio, además, facilita comprender el aporte del ambiente sobre estos y representa una forma de estudio novedosa para el uso de información de secuencias lineales. Siendo una metodología ampliamente utilizada para identificar residuos que aporten a la propiedad, por medio de la representación de señales asociadas al espacio de frecuencias [175, 51, 33].

A pesar de ser una metodología interesante a la hora de estudiar secuencias lineales, exhiben problemas notorios sobre la selección de las propiedades relevantes a analizar, ya que, existe un número considerablemente alto de propiedades posibles a utilizar, descritas principalmente en las base de datos AAIndex [99], y es factible que diferentes familias de proteínas exhiban comportamientos notoriamente no similares y diverjan en cuanto a las propiedades que puedan ser representativas, inclusive, a la hora de estudiar mutaciones en una misma proteína puede que no sólo una propiedad permita su caracterización, si no, que un conjunto pequeño de éstas [33].

En el presente capítulo, se exponen en detalle, diferentes formas de representar secuencias lineales de proteínas, seguido a su vez del planteamiento del uso de transformadas de Fourier para la digitalización de propiedades fisicoquímicas y cómo es posible utilizar éstas para la identificación de patrones en secuencias lineales o el desarrollo de modelos de clasificación/regresión y la exposición de casos de uso en diferentes proteínas de interés.

3.1 Metodologías asociadas a la codificación de variables categóricas

Diferentes metodologías existen para poder codificar variables categóricas, a su vez, para set de datos de proteínas con secuencias lineales, es factible utilizar sus propiedades fisicoquímicas o frecuencias de residuos. Las principales metodologías usadas a la fecha se expone a continuación.

3.1.1 One Hot encoder

One Hot encoder, es una de las técnicas más utilizadas a la hora de codificar variables categóricas y se basa principalmente en la adición de columnas con respecto a las categorías existentes en un conjunto de datos [32].

Dado el vector x de tamaño n con m categorías, por definición, One Hot encoder agrega al conjunto de datos m columnas, tal que, por cada categoría se adiciona una nueva columna al set de datos. Las nuevas columnas se completan con una binarización de los elementos, indicando si el elemento x_i posee la categoría m_j con un valor 1 y en caso contrario 0. Es posible expresar esto como se expone a continuación.

Sea x vector de m categorías de dimensiones $n \times 1$ representado por

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Su codificación mediante One Hot Encoder corresponde al vector x'(x)

$$x'(x) = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Esta metodología, si bien es altamente usada, implica que, a medida que aumentan la cantidad de categorías incrementan el número de columnas a agregar en el set de datos, es decir, si se tienen m categorías se adicionan m columnas. Esto último, puede provocar que los set de datos se afecten por problemas relacionados con la *maldición de dimensionalidad* [1], ya que, a medida que aumentan los descriptores, aumenta la probabilidad de que estos no sean informativos, provocando una adición de información innecesaria y que perjudicaría el rendimiento de los algoritmos de aprendizajes supervisado y no supervisado.

3.1.2 Ordinal encoder

Ordinal encoder, es una simplicación de One Hot encoder, ya que, simplemente codifica las categorías con números en el conjunto $[0, m-1]$. Es decir, sea el vector x de tamaño n con m categorías y sea M el espacio de las posibles categorías con $M = [m_1, \cdots, m_m]$, y

[1] Ver sección 2.1.11

cuya codificación implica el vector $M' = [0, \cdots, m-1]$. $\forall$ elemento que $\in$ a x se obtiene su codificación a partir del elemento $M'(M(m_i))$ que corresponde a la codificación de la categoría en el espacio M [140].

Es posible cuestionar esta metodología con respecto al orden en que trata las categorías, la representación de la información y el mantenimiento del significado de la data. Razón por la cual, se usa sólo en los casos en que la adición de múltiples descriptores empleando One Hot Encoder sea perjudicial a la hora de implementar modelos de clasificación o regresión, e inclusive, en la búsqueda de patrones.

3.1.3 Frecuencias de residuos

Una secuencia lineal de proteína, corresponde a un vector v de tamaño n donde cada elemento corresponde a un residuo que pertenece a la secuencia. El uso de esta información para alimentar modelos de clasificación o regresión conlleva la codificación de sus elementos. Sin embargo, a la hora de utilizar las codificaciones basadas en One Hot Encoder, el conjunto de datos no queda estándar en cuanto a sus dimensiones, ya que, el largo de las secuencias puede variar y a su vez, el número de columnas a agregar corresponde a $n \times 20$ dado a que son n residuos y el espacio muestral M es de tamaño 20 lo que genera un aumento considerable en la cantidad de dimensiones.

Con el fin de poder representar las secuencias lineales de proteínas, se idearon metodologías que consideran la frecuencia de aparición de los residuos en la secuencia, de tal manera, de poder codificarla en un vector de tamaño 20, donde cada elemento representa el número de incidencias del residuo dividido por el largo del vector. Así, cada elemento se encuentra en un rango $[0,1]$ donde 0 indica no incidencia del residuo y 1, incidencia total [136].

Expresado de forma matemática, sea s una secuencia lineal de proteínas con r residuos, su codificación se basa en la frecuencia de aparición del residuo en s, tal que, sea R el espacio de los posibles residuos r en s, se estima para cada $r_j \in R$ su frecuencia:

$$frec(r_j) = \frac{cont(r_j)\ if\ (r_i == r_j)}{n}$$

Finalmente, se tiene que cada residuo r_i se representa en su valor de frecuencia $frec(r_i)$, generando un set de datos de tamaños $s \times 20$ con s secuencias representadas por un vector de tamaño 20.

Esta es una de las representaciones más utilizadas y más simples a la hora de codificar secuencias lineales de proteínas. Sin embargo, presenta diferentes problemas tales como:

- Si los residuos se encuentran en proporciones similares, se generarán conjuntos de datos con atributos no informativos, ya que, disminuirá la varianza existente para dicho atributo, provocando una redundancia de datos.
- No considera información sobre los residuos asociados a propiedades fisicoquímicas, esto complica el hecho de representar un set de datos de secuencias o mutaciones, ya que no representa la realidad y sólo expone el comportamiento de las frecuencias de residuos, favoreciendo a aquellos con una mayor incidencia en sus elementos.
- La codificación por frecuencias es utilizada como un primer acercamiento a la representación del problema y principalmente perjudica a los modelos ya que puede generar atributos no informativos, como lo son los residuos sin incidencia, esto conlleva a modelos sobre ajustados y a creación de set de datos no informativos.
- No evalúa elementos relevantes a la caracterización de residuos claves, ambiente bajo el cual ocurren mutaciones o componentes adicionales que facilitarían una mayor comprensión del problema, ya que, sólo conocer las incidencias, proporciona un conocimiento sobre la moda y cuáles son los residuos más relevantes. No obstante, sólo permite inferir características, relacionadas a estos.

El uso de las frecuencias de residuos, es una de las primeras aproximaciones a la codificación de secuencias lineales de proteínas. No obstante, en todos los casos donde han sido utilizadas, se agrega información adicional, que permite comprender diferentes comportamientos y evalúa ciertas propiedades del entorno, razones por las cuales, se recomiendan utilizarlas en conjunto con otros descriptores.

3.1.4 Uso de propiedades fisicoquímicas

El uso de propiedades fisicoquímicas para describir un residuo, es ampliamente empleado en la generación de descriptores para conjuntos de datos en ingeniería de proteínas [36, 38]. Diversos enfoques y modelos han sido construidos o entrenados, contemplando información asociada a componentes termodinámicos del residuo, en particular, a la hora de describir residuos para evaluar cambios en la energía libre, relacionados a efectos en la estabilidad de una proteína [7, 31, 145].

Se han reportado cerca de 570 propiedades fisicoquímicas que pueden ser utilizadas para describir un residuo en una secuencia lineal de proteínas, almanceadas en la base de datos AAIndex [99]. A su vez, es posible caracterizar estos residuos empleando un conjunto de propiedades estructurales, termodinámicas e inclusive filogenéticas. Es decir, diferentes

puntos de vista que permitan describir los residuos pertenecientes a una secuencia. Sin embargo, el hecho de seleccionar qué descriptores son relevantes y cuáles no, radica en un problema de evaluación de características, el cual es común, en el área de la minería de datos.

Dado al gran conjunto de propiedades existentes y a la diversidad de descriptores que pueden ser utilizados para un conjunto de secuencias lineales de proteínas, es necesaria una selección correcta de las características, las cuales permitan formar set de datos informativos y con una correlación mínima entre sus elementos.

Contemplando esta problemática, técnicas de reducción de dimensionalidad o análisis de características son las más utilizadas a la hora de seleccionar los descriptores más informativos para un conjunto de datos. No obstante, en ocasiones, el conocimiento sobre el problema es un factor relevante a considerar.

Dada la relevancia de la selección de descriptores correctos para poder caracterizar y codificar secuencias lineales a partir de propiedades, se describen a continuación, algunas técnicas de reducción de dimensionalidad y análisis de características que pueden ser empleadas para dar solución a esta problemática.

Técnicas de reducción de dimensionalidad

Diferentes técnicas para el análisis de características y reducción de dimensionalidad han sido implementadas en el campo de minería de datos, con el fin de poder permitir la selección de descriptores informativos y sin contemplar conjuntos de datos altamente dimensionales. Dentro de las principales destacan: Análisis de correlación, mutual information, evaluaciones de características con respecto al entrenamiento de modelos empleando Random Forest, Análisis de componentes principales (PCA) y sus variantes como métodos lineales y reducción de dimensionalidad empleando métodos no lineales.

Análisis de correlación

La correlación entre elementos, es posible definirla como una relación estadística entre dos variables aleatorias, asociado principalmente, a relacionales lineales entre ellas [49]. Existen diferentes coeficientes de correlación, dentro de los cuales, el más conocido es Pearson. No obstante, se encuentran además: Spearman rank, kendall τ. La formulación matemática de estos, fue expuesta en la sección 2.1.10, en el apartado de análisis de desempeño de modelos de regresión.

De manera general, la correlación se relaciona con la covarianza entre las variables aleatorias [49], esto es: Sean X e Y variables aleatorias, el coeficiente de correlación ρ entre ellas se define como:

$$\rho X,Y = corr(X,Y) = \frac{cov(X,Y)}{\sigma_Y \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_Y \sigma_Y}$$

$\rho \in [-1,1]$ de tal forma que -1 indica una correlación inversa entre los elementos y 1 una relación positiva. En efectos prácticos, ambos valores denotan una dependencia entre los elementos, por lo que no son informativos y es posible eliminar uno de ellos, dada la redundancia de información. Valores cercanos a 0, indican que las variables no se encuentran correlacionadas.

Estos estudios se realizan para todas las variables en un conjunto de datos, expresándose en matrices de correlación, las cuales, pueden ser visualizadas mediante Heat Map en donde se expone la dependencia entre los atributos. Siendo una de las técnicas más utilizadas para la reducción de dimensionalidad, debido a su simpleza y a las ventajas que posee con respecto a la información que entrega sobre los atributos.

Mutual information

Es una medida de dependencia mutua entre dos variables aleatorias, permitiendo cuantificar la cantidad de información obtenida alrededor de una variable aleatoria vista desde otra. Representa un concepto más general que el análisis de correlación y se basa en la comparación de la similitud entre las distribuciones del conjunto de las variables independientes, con respecto al producto de éstas [141].

Sean dos variables aleatorias X e Y con valores en el espacio $X \times Y$ y sea el conjunto $P_{(X,Y)}$ y las distribuciones marginales son P_X y P_Y, respectivamente. El mutual information score para las variables X e Y es:

$$I(X,Y) = D_{KL}(P_{(X,Y)} || P_X \otimes P_Y)$$

Esto puede variar con respecto al tipo de variable a utilizar. En el caso de que pertenezcan a una distribución discreta el $I(X,Y)$ corresponde a:

$$I(X,Y) = \sum_{y_i \in Y} \sum_{x \in X} P_{(x,y)}(x,y) \log(\frac{P_{(x,y)}(x,y)}{P_x(x) P_y(y)})$$

Mientras que para el caso de variables con distribución continua se tiene:

$$I(X,Y) = \int_y \int_x P_{(x,y)}(x,y) \log(\frac{P_{(x,y)}(x,y)}{P_x(x) P_y(y)})$$

Como interpretación, se tiene que presenta un rango de valores no negativos, donde un valor 0 indica que no existe información mutua entre ambas características y a mayor valor más relación existe entre ambos elementos.

Análisis espaciales de características

En la sección 2.1.7, se describió Random Forest como un algoritmo de aprendizaje supervisado, que permite entrenar modelos utilizando árboles de decisión para manipular las características, generando *n* iteraciones, en donde se construyen *n* árboles, con diferentes atributos y ejemplos. Esto, permite estabilizar las medidas de desempeño y generar modelos robustos y con probabilidades menores de sobreajuste.

Sin embargo, este método puede ser utilizado con el fin de evaluar las características, ya sea en torno a cuáles son las frecuencias con las que permite dividir de manera inicial el conjunto de datos o cuáles son las que permiten generar la clasificación o la asignación a un intervalo en términos de regresión [154].

A partir de lo anterior, la profundidad de una característica utilizada como nodo de decisión en un árbol puede usarse para evaluar la importancia relativa de esa característica con respecto a la predictibilidad de la variable objetivo. Las características utilizadas en la parte superior del árbol contribuyen a la decisión de predicción final de una fracción mayor de las muestras de entrada. La fracción esperada de las muestras a las que contribuyen puede, por lo tanto, usarse como una estimación de la importancia relativa de las características [77].

La fracción de muestras a las que contribuye una característica se combina con la disminución de la impureza al dividirlas para crear una estimación normalizada del poder predictivo de esa característica [154].

Métodos de reducción de dimensionalidad lineales

Reducción de dimensionalidad, como su nombre lo sugiere, implica remover características o atributos del set de datos, con el objetivo de disminuir el número de dimensiones del conjunto de elementos. Esto permite descartar los atributos menos informativos o con menor relevancia, ya sea en términos de aporte a la varianza o relaciones con el resto de descriptores [87].

Dentro de las principales técnicas de reducción lineales, se encuentran el Análisis de componentes principales (PCA) y sus variantes, tales como: Incremental PCA [98].

Análisis de Componentes Principales (PCA, por sus iniciales en inglés), es una técnica estadística que permite la conversión de un conjunto de variables posiblemente correlacionadas a un conjunto de variables no correlacionadas linealmente. Estos elementos se denominan componentes principales. Su principal característica es que son ordenados de mayor a menor, según la varianza que entregan a los datos [98].

Intuitivamente, es posible pensar el PCA como un elipsoide n-dimensional de datos, donde cada eje del elipsoide representa un componente principal. Esto implica, que si algún

eje del elipsoide es pequeño, la varianza correspondiente a lo largo de éste también lo es, por lo que omitir dicho eje no implica una pérdida importante de información, lo cual es denotado como la reducción de la dimensionalidad en base a los aportes a las varianzas que denotan cada componente [181].

Definición

Matemáticamente, es posible definir PCA como una transformación lineal ortogonal de los datos a un nuevo sistema de coordenadas, tal que, la mayor varianza por alguna proyección de los datos pasa a situarse en la primera coordenada (llamado el primer componente principal), la segunda mayor varianza en la segunda coordenada, y así sucesivamente.

Se considera un conjunto de datos, X, con una media empírica 0, donde cada una de las filas representan ejemplos y las columnas características o atributos p.

La transformación está definida por un set de vectores de dimensión p que poseen pesos denotados por $w_{(k)} = (w_1, ..., w_p)_{(k)}$, los cuales para cada vector x_i en X se operan para dar un vector con los componentes principales $t_{(i)} = (t_1, ..., t_k)_{(i)}$ el cual viene dado por $t_{k(i)} = x_i w_k$

De tal manera que las variables individuales de t considerado sobre el conjunto de datos, sucesivamente heredan la varianza máxima posible de x, con cada carga del vector w.

El primer componente w_1 debe satisfacer las siguientes características:

- $w_1 = \arg\max_{||w||=1}\{\sum_i (t_{(1)})_i^2\} = \arg\max_{||w||=1}\{\sum_i (x_{(i)} * w)^2\}$
- $w_1 = \arg\max_{||w||=1}\{||Xw||^2\} = \arg\max_{||w||=1}\{w^T X^T X w\}$
- $w_1 = \arg\max\{\frac{w^T X^T X w}{w^T w}\}$

Los k restantes componentes son encontrados efectuando la extracción de los primeros *k-1* componentes principales desde x:

$$\hat{x}_k = x - \sum_{\delta=1}^{k-1} X w_{(\delta)} w_{(\delta)}^T$$

A su vez, para encontrar el vector de carga, es necesario extraer la varianza máxima del nuevo set de datos, tal que:

$$w_{(k)} = \arg\max_{||w||=1}\{||\hat{x}_k w||^2\} = \arg\max\{\frac{w^T X^T \hat{X}_k^T \hat{x}_k w}{w^T w}\}$$

La matriz de covarianza juega un rol fundamental en este análisis, cuyo valor entre dos componentes principales viene dado por:

$$Q(PC_j, PC_k) \infty (Xw_{(j)})^T * (Xw_{(k)})$$
$$Q(PC_j, PC_k) = w_{(j)}^T X^T X w_{(k)}$$
$$Q(PC_j, PC_k) = w_{(j)}^T \lambda_{(k)} w_{(k)}$$
$$Q(PC_j, PC_k) = \lambda_{(k)} w_{(j)}^T w_{(k)}$$

La principal característica que define al PCA es que es una técnica comúnmente utilizada para la reducción de la dimensionalidad, esto viene dado por la transformación que se genera, $T = xw$ donde cada vector $x_{(i)}$ existente en un espacio de coordenadas de variables p, es representado por un nuevo espacio, en el cual las variables no se encuentran correlacionadas. Sin embargo, si se utilizan L componentes principales para así utilizar los primeros L vectores de carga se obtiene una transformación truncada $T_L = XW_L$, de tal manera que la matriz T_L posee los n ejemplos originales.

No obstante, sólo posee L características que definen el set de datos, de tal manera que dicha transformación es posible expresarla como:

$t = W^T x$, donde $x \in R^p, t \in R^L$, para las cuales las columnas $p \times L$ de la matriz W forman una base ortogonal de las L características, de esta manera, al basarse en la construcción con sólo L columnas se maximiza la varianza original de los datos y se minimiza el error cuadrático tal que:

$$||TW^T - T_L W_L^T||_2^2 = ||X - X_L||_2^2$$

Normalmente esta reducción es usada para el manejo de set de datos de alta dimensionalidad.

PCA presenta algunas variaciones en sus definiciones o en el uso de los datos, tales como: Incremental PCA. Este método, representa una ventaja computacional en cuanto al coste de memoria y a la manipulación de set de datos altamente dimensionales [11].

Métodos de reducción de dimensionalidad no lineales

3.1.5 Codificación de residuos con adición de información de su entorno

Adicional a las técnicas explicadas previamente con respecto a las codificaciones existentes, en algunos casos, no sólo basta con una única codificación del residuo, si no, que es relevante adicionar información que puede ser importante para describir los residuos. Normalmente, junto con las codificaciones basadas en propiedades fisicoquímicas, se emplean técnicas que permitan describir el ambiente bajo el cual se encuentre el residuo [123].

En la gran mayoría de los casos, se adiciona información de los residuos cercanos al residuo de interés, esto depende del tipo de datos bajo el cual se esté trabajando, es decir, si son secuencias lineales o son estructuras de proteínas en formato PDB [38, 36].

Para el caso de que sean secuencias lineales, sea s secuencia de residuos de tamaño n y sea r_i el residuo de interés a evaluar su ambiente. Se crea una ventana de tamaño n' que contempla la cantidad de residuos r_j cercanos al residuo r_i, de tal manera que se crea un nuevo sub conjunto s' de datos de tamaño $2n'$ con n' residuos a la izquierda y n' a la derecha. El cual normalmente es codificado empleando binarización de elementos, así, en algunas ocasiones, a cada residuo, se le adicionan 20 descriptores que permiten indicar la ausencia o presencia de residuos cercanos a su entorno y el cual se completa con el conjunto de residuos s' [38].

Cuando se manejan estructuras de proteínas en formato PDB, la codificación y la evaluación del ambiente es similar. Sin embargo, en vez de utilizar una ventana de tamaño n' se utiliza un radio espacial de valor x para el cual, se toma el residuo y se estiman las distancias de los elementos cercanos, ya sea entorno a los carbonos α o a otros elementos. Esto, a diferencia de las secuencias lineales, permite adicionar información sobre las propiedades de distancia, ángulos y conformación de estabilidad por interacciones electrostáticas débiles que pueden generarse a partir de la proximidad de los elementos. No obstante, es una inferencia de su uso y se requieren de diferentes tipos de elementos que permitan caracterizar los eventos asociados al ambiente estructural asociado al residuo [38].

Actualmente, el uso de codificaciones mediante propiedades fisicoquímicas y el empleo de información adicional basada en descriptores de ambientes, es una de las metodologías más utilizadas a la hora de generar set de datos relacionados a mutaciones. Sin embargo, debido a que sólo se considera distancia, la binarización de los elementos no se ve afectada por sustituciones en residuos lejanos al lugar de ocurrencia, lo que denota la necesidad de idear metodologías que permitan contemplar el aporte completo de residuos a la caracterización de propiedades y cómo sustituciones puntuales afectan enormemente a residuos de interés. Una de las formas en las que se ha intentado dar solución a esta problemática, es modelar las propiedades fisicoquímicas de los residuos de las secuencias, a partir del uso de transformaciones de Fourier y en particular, empleando algoritmos relacionados a dichos conceptos, que aprovechen las ventajas referidas a la manipulación de espacios de frecuencias por sobre elementos temporales.

3.2 Transformaciones de Fourier

3.2.1 Transformada rápida de Fourier (FFT)

3.2.2 Uso de Transformadas de Fourier en digitalización de propiedades fisicoquímicas

3.3 Clustering

Clustering se define como un método de aprendizaje no supervisado, en el cual se cuenta con un conjunto de datos que representan a una muestra y en base a ésta, se trata de obtener grupos de objetos, denominados clusters.

Los clusters deben cumplir con dos características fundamentales:

- Los objetos que pertenezcan a un mismo clúster deben ser bastante homogéneos entre ellos.
- Entre los clústers debe existir un alto grado de heterogeneidad.

Los métodos de clustering se tratan, fundamentalmente, de resolver el siguiente problema: Dado un conjunto de N individuos, caracterizados por la información de n variables X_j con j entre $1,..,n$, se plantea el reto de ser capaces de agruparlos de manera que los individuos pertenecientes a un grupo (cluster), dada la información disponible, sean tan similares entre sí como sea posible, siendo los distintos grupos entre ellos tan disimilares como sea posible.

Básicamente, el análisis constará de un algoritmo de clusterización que permitirá la obtención de una o varias particiones, de acuerdo con los criterios establecidos.

3.3.1 Criterios de Similitud

Tal como se ha mencionado anteriormente, el hecho de tener elementos pertenecientes a un mismo grupo bastante similares entre ellos y divergentes entre distintos clúster, es la característica primordial a tratar, esto último radica en la importancia de las variables que componen a un elemento en particular y en los valores que tomen éstas.

Por lo tanto se debe determinar que tan similares o diferentes son los valores que tomen las variables con respecto al elemento al cual pertenece.

Para medir lo similar o disimilar que son los individuos existe una enorme cantidad de índices de similaridad y de disimilaridad o divergencia. Todos ellos tienen propiedades y utilidades distintas y habrá que ser consciente de ellas para su correcta aplicación al momento de hacer uso de una de ellas.

Los índices expuestos normalmente pueden ser clasificados tal como sigue:

1. Indicadores basados en la distancia, para lo cual se considera a los individuos como vectores en el espacio de las variables, en este sentido un elevado valor de la distancia entre dos individuos indicará un alto grado de disimilaridad entre ellos.

2. Indicadores basados en coeficientes de correlación; la correlación permite indicar cuál es la fuerza y la dirección de una relación lineal entre dos variables. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores de la otra, esto es: si se tiene dos variables (A y B) existe correlación si al aumentar los valores de A lo hacen también los de B y viceversa.

3. Indicadores basados en tablas de datos de posesión o no de una serie de atributos, es decir, teniendo dos vectores A y B que representan a dos individuos de una población se hace una comparación de los elementos existentes en A y no en B, los elementos existentes en B y no en A además de la intersección entre ellos, es decir, los elementos existentes en A y en B.

3.3.2 Algoritmos de Clustering

Existen diversos algoritmos de clustering, cada uno con características que los diferencian, los cuales, pueden ser aplicados a diversos casos, dependiendo de las características de los datos de entrada, es decir, de la geometría de estos datos. Sin embargo, esta representación se basa principalmente en el uso de matrices, donde cada fila representa un ejemplo y cada columna el valor de un atributo o rasgo cualitativo para dicho ejemplo.

A continuación, se resumen algunos de los algoritmos de clustering más utilizados, explicando sus propiedades y sus características.

k-Means

El algoritmo *k*-Means, trata la separación de muestras en n grupos de igual varianza, minimizando el criterio conocido como inercia, lo que se traduce en la suma de los cuadrados dentro de los clúster.

La principal característica y deficiencia a la vez, es que se requiere que el número de grupos sea entregado, es decir, se debe entregar el valor de k, así, si se selecciona un valor de $k = 3$, serán tres grupos los que se encontrarán.

Este algoritmo divide un set de N ejemplos X en K particiones distintas denominadas clúster C, cada uno de ellos descrito por la media μ_j de las muestras en el clúster. Esta media

es llamada centroide, por lo que en general, *k*-means elige sus centroides de tal manera que el principio de inercia sea reducido al mínimo, es decir, que la suma de los cuadrados de los integrantes de un mismo grupo sea mínima, a través de:

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

Sin embargo, el principio de inercia, o la suma de los cuadrados mínimos entre los integrantes de los clustering, puede sufrir varios inconvenientes:

- Se hace la suposición de que las agrupaciones son convexas e isotrópicas, lo cual no se da siempre, razón por la que responde mal ante a clusters que posean forma alargadas o con formas irregulares.
- No es una métrica normalizada, es decir, se sabe que los valores más bajos son mejores y el cero es óptimo. Sin embargo, en espacios de muy de alta dimensionalidad, las distancias euclidianas tienden a ser infladas, lo que se conoce como "la maldición de la dimensionalidad", razón por la cual, son utilizados algoritmos de reducción de la dimensionalidad, tal como PCA.

Ambos puntos, son posibles observarlos en la Figura 3.1, en la cual se exponen, problemas con varianzas distintas, diferencias asociadas al tamaño de los clúster, anisotropía[2] de los datos, etc.

En términos básicos, el algoritmo tiene tres pasos. El primer paso consiste en elegir los centroides iniciales, con el método más básico para elegir *k* muestras del conjunto de datos *X*. Después de la inicialización, *k*-means consta de un bucle entre los otros siguientes dos pasos.

En una primera instancia, asigna cada muestra a su centroide más cercano. Posterior a ello, se crean nuevos centroides tomando el valor medio de todas las muestras asignadas a cada centroide anterior. La diferencia entre la media anterior y la actual (diferencia entre centroides) se calcula y se itera estas acciones hasta que este valor sea inferior a un umbral. En otras palabras, se repite hasta que los centroides no se mueven de manera significativa.

Clustering Jerarquizado

Clustering jerárquico o HCA (por sus siglas en inglés) es un método de análisis de conglomerados, que busca construir una jerarquía de agrupaciones. Las estrategias para la agrupación son posible dividirlas en dos:

[2]Las variables varían en base a las direcciones en las que se examinan

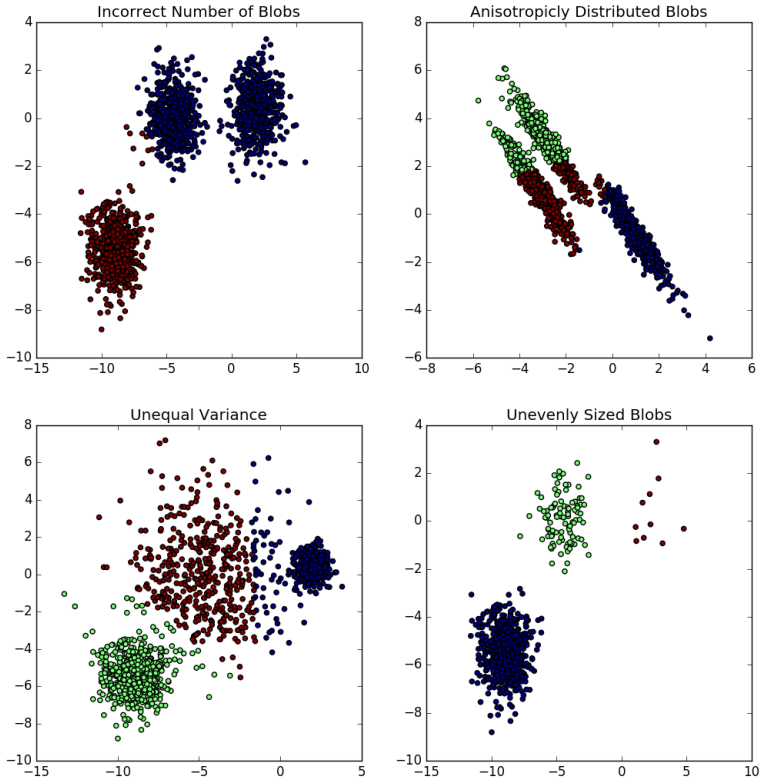


Fig. 3.1 Posibles inconvenientes con los datos, donde k-medias no funciona correctamente

- **Aglomerativa**: consiste en un enfoque de "abajo hacia arriba": cada observación se inicia en su propio clúster, y pares de grupos se fusionan a medida que se asciende en la jerarquía.
- **Divisiva**: consiste en un enfoque "de arriba hacia abajo": todas las observaciones se inician en un único clúster, y las divisiones se realizan de forma recursiva conforme se desciende en la jerarquía.

Siendo generalmente expuestos los resultados en forma de dendrograma, por otro lado, la complejidad del algoritmo, en el caso general es $O(n^2 \log(n))$, lo cual presenta problemas para set de datos extensos.

Con el fin de decidir qué grupos se deben combinar (por aglomeración), o cuando un grupo se debe dividir (por división), se requiere una medida de disimilitud entre los conjuntos de observaciones. En la mayoría de los métodos de agrupación jerárquica, esto se logra mediante el uso de una métrica apropiada (una medida de la distancia entre pares de

observaciones), y un criterio de vinculación que especifica la disimilitud de conjuntos como una función de las distancias por pares de observaciones en los conjuntos.

Las métricas asociadas pueden son las mismas utilizadas en el algoritmo *k*-means y las aplicadas en el método KNN de aprendizaje supervisado.

El criterio de linkage determina la distancia entre conjuntos de observaciones como una función de las distancias por pares entre observaciones.

Algunos criterios de linkage de uso común entre los dos conjuntos de observaciones A y B son expuestos en la tabla 3.1.

Criterios de linkage comunes en métodos HCA	
Complete linkage clustering	$\max\{d(a,b) : a \in A, b \in B\}$.
Single-linkage clustering	$\min\{d(a,b) : a \in A, b \in B\}$.
Average linkage clustering	$\frac{1}{\|A\|\|B\|}\sum_{a\in A}\sum_{b\in B} d(a,b)$.
Centroid linkage clustering	$\|c_s - c_t\|$ con c_s y c_t centroides
Minimum energy clustering	$\frac{2}{nm}\sum_{i,j=1}^{n,m}\|a_i - b_j\|_2 - \frac{1}{n^2}\sum_{i,j=1}^{n}\|a_i - a_j\|_2 - \frac{1}{m^2}\sum_{i,j=1}^{m}\|b_i - b_j\|_2$.

Table 3.1 Resumen de linkages comunes utilizados en métodos de clustering jerárquicos.

Un aspecto interesante de este algoritmo es que pueden ser añadidas las limitaciones de conectividad, es decir, sólo grupos adyacentes pueden fusionarse entre sí, esto es, a través de una matriz de conectividad que define para cada muestra, las muestras de vecinos después de una estructura dada de los datos. Estas restricciones son útiles para imponer una cierta estructura local, así como para hacer que el algoritmo sea más rápido, especialmente cuando el número de las muestras es alta.

Affinity Propagation

AffinityPropagation crea grupos mediante el envío de mensajes entre pares de muestras hasta la convergencia. Un conjunto de datos es descrito por el uso de un pequeño número de ejemplares, que se identifican como las más representativas de otras muestras. Los mensajes enviados entre pares representan la idoneidad para una muestra a ser el ejemplo de la otra, la cual se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de forma iterativa hasta la convergencia, momento en el que se eligen los ejemplares finales, y por lo tanto se da el agrupamiento final.

Se elige el número de grupos en base a los datos proporcionados. Para este propósito, los dos parámetros importantes son la preferencia, que controla el número de ejemplares que se utilizan, y el factor de amortiguamiento.

El principal inconveniente que presenta este algoritmo viene dado por la complejidad que posee, el cual se representa por $O(N^2T)$, donde N es el número de muestras y T es el número de operaciones necesarias para converger, razón por la cual, el uso de este algoritmo es para set de datos con pequeña cantidad de ejemplos.

Con respecto a la descripción del algoritmo, es posible mencionar que los mensajes enviados a los grupos, pertenecen a dos categorías, la primera es la responsabilidad $r(i,k)$ la cual consiste en la evidencia acumulada que denota que la muestra k podría ser un ejemplar para la muestra i. La segunda es la disponibilidad $a(i,k)$, la cual se define como la evidencia existente para que la muestra i pueda escoger a la muestra k para ser su ejemplar, además considera todos los valores de las otras muestras de k que podrían ser ejemplares. De esta manera, los ejemplares son escogidos por las muestras si:

- Existe una similaridad bastante alta con respecto a las muestras.
- Si es elegido por muchas muestras y resulta ser representativo de sí mismos.

En forma matemática es posible definir la responsabilidad como:

$$r(i,k) \leftarrow s(i,k) - max[a(i,\acute{k}) + s(i,\acute{k}) \forall \acute{k} \neq k]$$

Donde $s(i,k)$ es la similaridad entre las muestras i y k.
A su vez, la disponibilidad, es posible definirla como:

$$a(i,k) \leftarrow min[0, r(k,k) + \sum_{\acute{i}\ s.t.\ \acute{i} \notin \{i,k\}} r(\acute{i},k)]$$

Mean Shift

El algoritmo de Mean Shift tiene como objetivo descubrir manchas (blobs) en una densidad uniforme de las muestras. Es un algoritmo basado en centroides, que funciona mediante la actualización de los candidatos para centroides para ser la media de los puntos dentro de una región determinada. Estos candidatos se filtran en una etapa de post-procesamiento para eliminar duplicados y así formar el conjunto final de centroides.

Dado un candidato x_i para la iteración t, el candidato a centroide es actualizado en base a la ecuación:

$$x_i^{t+1} = x_i^t + m(x_i^t)$$

Donde $N(x_i)$ es la vecindad de las muestras dentro de una distancia dada alrededor x_i y m es el vector de desplazamiento medio, que se calcula para cada centroide que apunta hacia una región del aumento máximo en la densidad de puntos. Ésta se calcula utilizando la siguiente ecuación, en la que la actualización de manera efectiva denota a un centroide ser la media de las muestras dentro de su vecindad:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

Algunas de las características de Mean Shift, son

- Ajusta automáticamente el número de grupos, dependiendo de un parámetro bandwidth, el cual se asocia al tamaño de la región en la que se debe buscar los centroides.
- No es altamente escalable, ya que requiere múltiples búsquedas de vecinos más cercanos durante la ejecución del algoritmo.
- Garantiza convergencia, deteniendo la iteración cuando el cambio en centroides es pequeño.

DBSCAN

El algoritmo DBSCAN ve agrupaciones como áreas de alta densidad separadas por zonas de baja densidad. Debido a esta visión bastante genérica, las agrupaciones que se encuentran pueden ser de cualquier forma, en lugar de *k*-means que supone que los grupos tienen la forma convexa.

El componente central de DBSCAN es el concepto de muestras de núcleo, las cuales son las muestras que se encuentran en áreas de alta densidad. Por lo tanto, un clúster es un conjunto de muestras de núcleos, cada uno cerca del otro (medido por alguna medida de distancia) y un conjunto de muestras no básicas que se encuentran cerca de una muestra básica. Hay dos parámetros necesarios para el algoritmo, min samples y EPS, los cuales definen formalmente la densidad deseada.

Más formalmente, se define una muestra del núcleo como una muestra del conjunto de datos de tal manera que existe una cantidad de muestra mínimas y a su vez otras muestras dentro de una distancia de EPS, que se definen como vecinos de la muestra del núcleo. Esto dice que la muestra de núcleo se encuentra en un área densa del espacio vectorial. Un clúster es un conjunto de muestras de núcleo que se puede construir mediante la adopción de forma recursiva de una muestra básica, la búsqueda de todos sus vecinos que son muestras de la base, la búsqueda de la totalidad de sus vecinos que son muestras de núcleo, y así sucesivamente. Un cluster también tiene un conjunto de muestras no básicas, que son las muestras que son vecinos de una muestra básica de la agrupación, pero no son en sí mismos muestras de núcleos. Intuitivamente, estas muestras están al margen de un clúster.

Cualquier muestra de núcleo es parte de un clúster, por definición. Cualquier muestra que no es una muestra del núcleo, y está al menos una distanca eps de cualquier muestra del núcleo, se considera un valor atípico por el algoritmo.

Normalmente, los resultados del algoritmo, pueden representarse tal como se expone en la Figura 3.2, el color indica la pertenencia al clúster, con grandes círculos que indican muestras de núcleos encontrados por el algoritmo, círculos más pequeños son muestras no básicas que todavía son parte de un clúster. Por otra parte, los valores atípicos se indican con puntos negros.

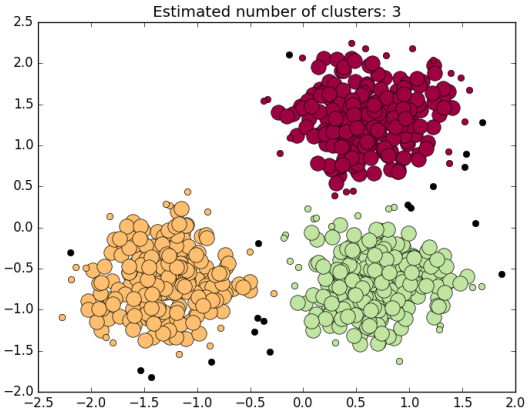


Fig. 3.2 Representación de resultados al aplicar la clusterización por DBSCAN

Birch

Birch construye un árbol llamado Characteristic Feature Tree (CFT) de los datos correspondientes. Los datos están esencialmente con pérdida de información, comprimidos en un conjunto de nodos de rasgo característico denominados CF nodos. Estos tienen una serie de subgrupos llamados subclusters de rasgo característico, ubicados en los nodos CF no terminales los cuales pueden tener CF nodos como hijos.

Los subgrupos CF pueden contener la información necesaria para la agrupación que evite la necesidad de mantener los datos de entrada enteros en la memoria. Esta información incluye:

- Número de muestras en un subgrupo.
- Suma lineal, representada por un vector n-dimensional que sostiene la suma de todas las muestras.
- Suma al cuadrado, representada por la suma cuadrática de la norma L2 de todas las muestras.

- Centroides, para evitar un nuevo cálculo de sumas lineales con respecto al número de muestras.

- Norma al cuadrado de los centroides.

El algoritmo de Birch tiene dos parámetros, el umbral y el factor de branching. El factor de branching limita el número de subgrupos en un nodo y el umbral limita la distancia entre la muestra de entrada y los subclusters existentes.

Este algoritmo puede ser visto como un método de instancia o reducción de datos, ya que reduce los datos de entrada a un conjunto de subclusters que se obtienen directamente de las hojas de la CFT. Estos datos reducidos pueden ser procesados por la alimentación en un clúster global. Este clúster global puede ser establecido por n clusters. Si este parámetro se establece como valor 0 o ninguno, los subgrupos de las hojas se leen directamente, de lo contrario un paso global de la agrupación etiqueta estos subgrupos en grupos globales y las muestras se asignan a la etiqueta global del subgrupo más cercano.

Una descripción del algoritmo, es posible realizarla en los siguientes puntos:

- Una nueva muestra se inserta en la raíz del árbol CF que es un nodo CF. A continuación, se fusiona con el subgrupo de la raíz, el que tiene el radio más pequeño después de la fusión, limitada por el umbral de ramificación y condiciones de los factores. Si el subcluster tiene algún nodo hijo, entonces esto se realiza repetidamente hasta que llega a una hoja. Después de encontrar el subcluster más cercano en la hoja, las propiedades de este subgrupo y los subclusters padres se actualizan de forma recursiva.

- Si el radio del subcluster obtenido mediante la fusión de la nueva muestra y el subgrupo más cercano es mayor que el cuadrado del umbral y si el número de subclusters es mayor que el factor de ramificación, a continuación, un espacio se asigna temporalmente a esta nueva muestra. Los dos subgrupos más lejanos se toman y de los subgrupos se dividen en dos grupos sobre la base de la distancia entre estos subgrupos.

- Si este nodo de división tiene un subgrupo de los padres y no hay espacio para un nuevo subgrupo, entonces el padre se divide en dos. Si no hay espacio, entonces este nodo se divide de nuevo en dos y el proceso se continúa de forma recursiva, hasta que llega a la raíz

Mixture Model

Los métodos de clustering basado en modelos tratan de optimizar el conjunto de datos a un modelo matemático. En general estos métodos se basan en la suposición que los datos

han sido generados por una mezcla de distribuciones de probabilidad. Dentro de los más utilizados se encuentran **Gaussian Mixture** y **Expectation-Maximization**.

Expectation Maximization, supone que los datos emergen de una mecla de distribuciones, donde cada distribución se denomina como *component distribution*, razón por la cual, los datos pueden agruparse usando un modelo de mezcla de densidades de k distribuciones de probabilidades. Sin embargo, el problema reside en estimar los parámetros de estas distribuciones para proveer del mejor ajuste posible a los datos.

El algoritmo Expectation Maximization (EM) puede ser considerado como una extensión de k-means, esto es debido a que: Si k-means asigna cada objeto a un clúster en función de su media, EM asigna cada objeto a un clúster en función de un peso que representa la probabilidad de pertenencia al clúster. Esto requiere que se defina una distribución de probabilidad para los clusters.

Por otro lado, un modelo de mezcla gaussiano como Gaussian Mixture Model (GMM) es una función de densidad de probabilidad representada por una suma de componentes gausianas, GMMs son usadas como modelos paramétricos de la distribución de probabilidad de medidas continuas, donde los parámetros de GMM son estimados usando iterativamente el algoritmo Expectation-Maximization.

Un GMM es una suma con pesos de densidades gaussianas:

$$p(\vec{x}) = \sum_{i=1}^{M} w_i \times N(\vec{x}|\mu_i, \sum_i)$$

Donde $\vec{x}$ es un vector D-dimensional de datos, w_i son los pesos con $\sum_{i=1}^{M} w_i = 1$, y $N(\vec{x}|\mu_i, \sum_i)$ es la densidad gaussiana, por lo tanto, la caracterización se completa con la media, la matríz de covarianza y el peso de cada componente gaussiana.

Una de las limitantes es que el número de componentes gausianos tiene que ser fijado al principio del algoritmo.

El GMM consta de los siguientes pasos:

1. **Iniziacilización**: para cada clase, un vector compuesto de la media y la matriz de covarianza es construido. Este vector representa las características de la distribución gaussiana usada para caracterizar las entidades del conjunto de datos. Inicialmente estos valores son generados aleatoreamente, posteriormente el algoritmo EM trata de aproximar los valores del vector de la distribución real de los datos.

2. Se estima la probabilidad de cada elemento de pertenecer a un clúster.

3. Se estiman los parámetros de la distribución de probabilidad para el próximo ciclo, primero se calcula la media de la clase a través de la media de todos los puntos en

función del grado de relevancia de cada punto, continuando con el cálculo de la matriz de covarianza.

4. **Convergencia**: Después de cada ciclo se ejecuta un test de convergencia para verificar cuánto ha cambiado el vector de parámetros y si la diferencia es menor que un umbral de tolerancia el algoritmo se detiene, no obstante es posible detener el algoritmo debido al alcance de un número máximo de ciclos.

Una representación visual del modelo es posible observarla en la Figura 3.3, en ella se aprecia cómo a medida que se va iterando el algoritmo se generan los cambios y las *separaciones* en grupos de clúster definidos.

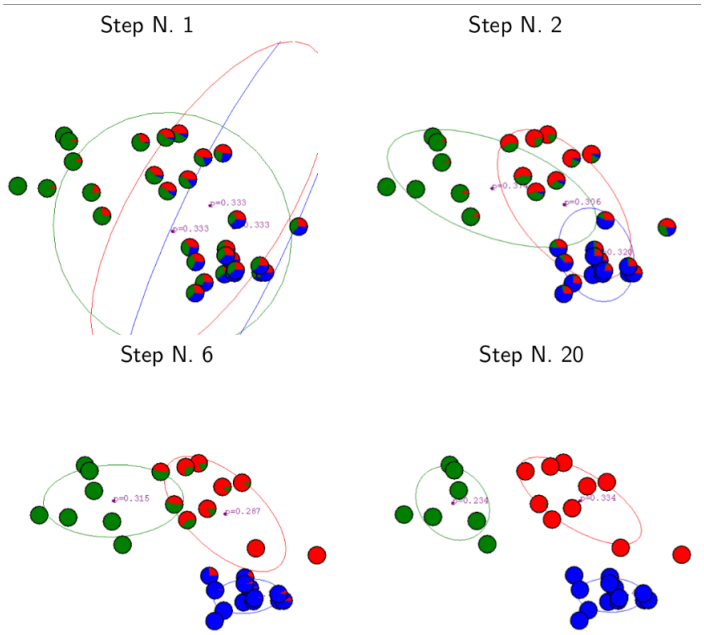


Fig. 3.3 Esquema representativo de cambios durante las iteraciones en GMM

Cuadro Resumen

En la Tabla 3.2 se expone un resumen de las características de cada algoritmo expuesto, la escalabilidad que poseen, las distancias que ocupan, los casos de uso y los parámetros que poseen.

Evaluación del desempeño de un clustering

Evaluar el desempeño de un algoritmo de clustering no es tan trivial como contar el número de errores o la precisión y la recuperación de un algoritmo de clasificación supervisada.

Tabla resumen de Algoritmos de Aprendiza No Supervisado				
Algoritmo	**Parámetros**	**Escalabilidad**	**Usos**	**Métrica usada**
K-Means	Número de clúster	Muchas muestras, mediana cantidad de clúster.	De propósito general, la geometría plana, no demasiados grupos	Distancia entre puntos
Affinity propagation	preferencia	No escalable con n ejemplos	Muchos clúster, tamañode clúster desigual, geometría no plana	Distancia gráfica
Mean-shift	bandwidth	No escalable con n ejemplos	Muchos clúster, tamañode clúster desigual, geometría no plana	Distancia entre puntos
Ward hierarchical clustering	Número de clúster	Mucha cantidad de ejemplos y de clusters	Cualquier clúster, es posible conección de constraints	Distantia entre puntos
Agglomerative clustering	Número de clúster, tipo de unión, distancia	Mucha cantidad de ejemplos y de clusters	Muchos clusters, posiblemente restricciones de conectividad, distancias no euclidianas	Cualquier distancia pairwise
DBSCAN	tamaño vecino	Mucha cantidad de ejemplos, mediana cantidad de clúster	Geometría no plana, tamaños de clusters distintos	Distancia entre puntos vecinos
Gaussian mixtures	variado	No escalable	Geometría plana, bueno para la estimación de la densidad	Distancia Mahalanobis para los centros
Birch	branching, umbral	Alto número de clúster y ejemplos	Largo set de datos, eliminación valores atípicos, reducción de datos	distancia euclidiana entre puntos

Table 3.2 Cuadro resumen de algoritmos de aprendizaje supervizado

En particular, cualquier métrica de evaluación no debe tomar los valores absolutos de las etiquetas de clúster en cuenta, sino más bien si estas agrupaciones definen separaciones de los datos, de tal manera que los miembros que pertenecen a la misma clase son más similares que los miembros de diferentes clases de acuerdo con alguna similitud métrica.

Existen diversas medidas de similitud con el fin de evaluar el clustering, las cuales se explican a continuación:

Índice Rand ajustado (Adjusted Rand index)

Dado el conocimiento de las clases asignadas como verdaderas (etiquetas verdaderas) y las etiquetas obtenidas por el algoritmo de clustering (etiquetas predichas) el adjuster rand index es una función que mide la similaridad de las dos asignaciones, ignorando permutaciones, posee valores entre -1 y 1, siendo 1 el valor perfecto. Sin embargo, es imperante para evaluar el desempeño, conocer las etiquetas verdaderas de los datos.

Matemáticamente es posible definirlo como:

Sea C una asignación de clase real y dada la agrupación K, se define a y b como:

- a, el número de pares de elementos que estan en el mismo set en C y en el mismo set en K.
- b, el número de pares de elementos que estan en diferentes set en C y en diferentes set en K.

El valor del rand index no ajustado viene dado por:

$$RI = \frac{a+b}{C_2^{n_{samples}}}$$

Donde $C_2^{n_{samples}}$ es el número total de posibles pares en el set de datos.

Sin embargo, la puntuación de RI no garantiza que las asignaciones de etiquetas al azar conseguirán un valor cercano a cero, para contrarrestar este efecto se puede descartar la esperanza RI $E[RI]$ de etiquetas al azar mediante la definición del adjusted rand index:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

Información mutua basada en scores

Dado el conocimiento de las etiquetas de las clases reales y las asignaciones obtenidas de algoritmos de agrupación de las mismas muestras, el mutual information es una función que

mide el *acuerdo* de las dos asignaciones, ignorando las permutaciones. Existen dos versiones normalizadas diferentes de esta medida:

Normalized Mutual Information, NMI (Información mutua normalizada) y Adjusted Mutual Information, AMI (Información mutua ajustada). NMI es a menudo usado en la literatura mientras que AMI fue porpuesto más recientemente.

Matemáticamente, es posible definir esta forma de evaluación tal que: se asume dos etiquetas asignadas (de los mismos N objetos), U y V, su entropía es la cantidad de incertidumbre para un conjunto de particiones definido por:

$$H(U) = \sum_{i=1}^{|U|} P(i)\log(P(i))$$

donde $P(i) = |U_i|/N$ es la probabilidad que un objecto seleccionado aleatoriamente de la clase U sea asignado a la clase U_i, de igual manera para V:

$$H(V) = \sum_{j=1}^{|V|} P'(j)\log(P'(j))$$

Con $P'(j) = |V_j|/N$ el mutual information (MI) entre U y V es calculado por:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log(\tfrac{P(i,j)}{P(i)P'(j)})$$

donde $P(i,j) = |U_i \cap V_j|/N$ es la probabilidad de que un objeto seleccionado aleatoriamente sea asignado a ambas clases U_i y V_j.

El valor normalizado del mutual information es definido como:

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}$$

Este valor del mutual information y también la variante normalizada no se ajusta al azar y tiende a aumentar a medida que aumenta el número de diferentes etiquetas (clusters), independientemente de la cantidad real de *mutual information* entre las asignaciones de etiquetas.

El valor esperado para el mutual information puede ser calculado usando la ecuación descrita por Vinh, Epps, and Bailey, (2009). En esta ecuación, $a_i = |U_i|$ (el número de elementos en U_i) y $b_j = |V_j|$ (el número de elementos en V_j).

$$E[MI(U,V)] = \sum_{i=1}^{|} U|\sum_{j=1}^{|} V|\sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log(\frac{N.n_{ij}}{a_i b_j})\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}$$

Usando el valor esperado, el adjusted mutual information puede ser calculado usando una forma similar al ARI:

$$AMI = \frac{MI-E[MI]}{\max(H(U),H(V))-E[MI]}$$

Homogeneidad, Completación y medida V (V-measure)

Para el caso en el que se conozca a ciencia cierta las etiquetas reales de las clases, es posible definir medidas de evaluación basándose en la entropía existente. En particular Rosenberg y Hirschberg (2007) [**?**] definen los siguientes dos objetivos deseables para cualquier asignación de clusters:

- **homogeneidad**: cada clúster contiene sólo miembros de una única clase.
- **Totalidad (completeness)**: todos los miembros de una clase son asignados a un mismo clúster.

Los valores de estos score abarcan los rangos entre 0 y 1, siendo 1 el score perfecto.
Es posible definir la homogeneidad y el completeness como:

$$h = 1 - \frac{H(C|K)}{H(C)}$$
$$c = 1 - \frac{H(K|C)}{H(K)}$$

donde $H(C|K)$ es la entropía condicional de las clases dada la asignación del clúster y es definida por:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log(\frac{n_{c,k}}{n_k})$$

y $H(C)$ es la entropía de la clase y cuyo valor viene dado por:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot\log(\frac{n_c}{n})$$

con n siendo el número total de muestras, n_c y n_k el número de muestras respectivamente pertenecientes a la clase c y al clúster k, y finalmente $n_{c,k}$ el número de muestra de la clase c asignados al clúster k.

Rosenberg y Hirschberg también definieron un **V-measure** como el score medio de la homogeneidad y completeness:

$$v = 2\cdot\frac{h\cdot c}{h+c}$$

Coeficiente de silueta (Silhouette Coefficient)

Este coeficiente es posible utilizarlo cuando se desconocen las reales etiquetas de los ejemplos, una puntuación alta (por sobre 0.75) denota un modelo con grupos bien definidos. Este coeficiente se define para cada muestra y posee dos score:

- **a**: la distancia media entre un ejemplo y todos los otros puntos en la misma clase.
- **b**: la distancia media entre un ejemplo y todos los otros puntos en el siguiente clúster vecino.

El coeficiente para una única muestra, viene dado por:

$$s = \frac{b-a}{max(a,b)}$$

Calinski-Harabaz Index

Este índice es utilizado cuando las etiquetas son desconocidas, donde un mayor valor de éste implica un modelo mejor definido.

Para k clusters, el Calinski-Harabaz index s se da como la razón de la dispersión entre clusters y la dispersión dentro del grupo:

$$s(k) = \frac{Tr(B_k)}{Tr(W_k)} \frac{N-k}{k-1}$$

Donde B_K es la matriz de dispersión entre grupos y W_K es la matriz de dispersión dentro del clúster definida por:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$
$$B_k = \sum_q n_q (c_q - c)(c_q - c)^T$$

Con N como el de puntos en el set de datos, C_q el set de puntos en el cluster q, c_q el centro del clúster q, c el centro de E, n_q el número de puntos en el clúster q.

3.4 Hipótesis

Dada la problemática existente sobre cómo representar conjuntos de secuencias lineales con el fin de poder desarrollar modelos de clasificación/regresión o identificación de patrones asociados a residuos claves que brindan la propiedad fisicoquímica. Y, en consideración de los diferentes usos que entrega las transformadas de Fourier, se plantea la hipótesis de este capítulo.

La codificación de secuencias lineales empleando espectros de frecuencia, basados en las propiedades fisicoquímicas de los residuos pertenecientes, permite generar descriptores que faciliten el aprendizaje de predictores de variantes enfocados a diferentes respuestas de interés? A su vez, los patrones de frecuencia, pueden ser asociados a residuos claves en familias de variantes?

Si bien, en el planteamiento de la hipótesis se exponen dos preguntas, la interrogante en sí, se centra a los posibles usos que pueda tener los espectros de frecuencia en el estudio de variantes, identificación de patrones, residuos relevantes, etc.

3.5 Objetivos

En base a la hipótesis planteada y con el fin de responder a los planteamientos e interrogantes expuestas. Se detallan el objetivo general y los objetivos específicos.

3.5.1 Objetivo general

Diseñar e implementar metodología de codificación y digitalización de propiedades fisicoquímicas en secuencias lineales de proteínas, con el fin de poder ser utilizadas en identificación de patrones por medio de técnicas de clustering o desarrollo de predictores basados en algoritmos de aprendizaje supervisado.

3.5.2 Objetivos específicos

A partir del objetivo general, nacen los siguientes objetivos específicos.

1. Preparar y manipular base de datos de propiedades fisicoquímicas asociadas a la base de datos AAindex [99].

2. Diseñar e implementar, metodología de codificación de propiedades fisicoquímicas y selección de las más representativas, por medio de técnicas de reducción de dimensionalidad y selección de features y descritas a través de espectros de frecuencia.

3. Implementar y validar modelos de clasificación para evaluación de análisis de estabilidad de variantes según descriptores basados en espectros de frecuencia de propiedades fisicoquímicas.

4. Diseñar, implementar y validar metodología para identificación de patrones asociados a residuos y la generación de clustering de espectros de frecuencia.

3.6 Metodología

Con el fin de poder cumplir con el objetivo general planteado y los objetivos específicos, se expone a continuación la metodología diseñada. Se consideran diferentes etapas dentro de las cuales se destaca la codificación, entrenamiento de modelos, aplicación de clustering e identificación de residuos como patrones de señales dentro del espectro de frecuencia.

A continuación de exponen las diferentes etapas asociadas al proceso.

3.6.1 Codificación de secuencias lineales

La codificación de secuencias lineales, se basa en el uso de propiedades fisicoquímicas representativas de la secuencia, las cuales se obtienen a partir de la base de datos AAindex [99].

Un esquema representativo del proceso, se observa en la Figura 3.4, en la cual, se detallan los diferentes pasos a seguir para generar la codificación correspondiente y obtener los espectros de frecuencias asociados a cada propiedad fisicoquímica.

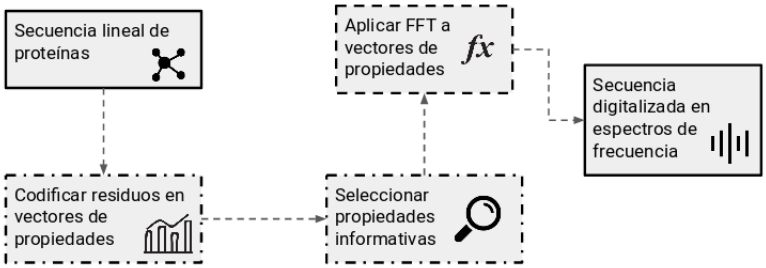


Fig. 3.4 Esquema representativo, metodología de digitalización de secuencias.

En una primera instancia, se toma la secuencia y por cada residuo se crea un vector de tamaño n el cual representa el número de propiedades fisicoquímicas descritas en la base de datos AAindex. De esta forma, se crea una matriz de tamaño $r \times n$ donde r representa la cantidad de residuos en la secuencia.

A partir de dicha matriz, técnicas de reducción de dimensionalidad y selección de características son implementadas, utilizando lenguaje de programación Python y la librería scikit-learn [140], con el fin de seleccionar cuáles son las propiedades más representativas y qué porcentaje de la varianza permiten explicar.

Dado el conjunto de propiedades seleccionadas, se implementarán rutinas basadas en lenguaje de programación Matlab, las cuales reciben el conjunto inicial de datos, en una primera instancia, aplica "zero-padding" con el fin generar vectores de tamaño 1024, requisito para la aplicación de FFT. A partir de esto, cada columna en el conjunto de elementos, se

digitaliza por medio del uso de la transformada rápida de Fourier (FFT) y se obtienen los espectros de frecuencias para cada propiedad fisicoquímica seleccionada previamente.

De esta forma, por cada secuencia, se obtiene un conjunto de espectros de frecuencia, asociados a la digitalización de las propiedades fisicoquímicas seleccionadas mediante técnicas selección de características.

3.6.2 Implementación de modelos de clasificación/regresión para análisis de variantes

Uno de los objetivos de la codificación de secuencias lineales, es evaluar si el conjunto de espectros de frecuencia para un grupo de variantes, puede ser utilizado como características para generar set de datos y entrenar modelos a partir de estos.

Con esto en mente y apoyados en los conjuntos de datos utilizados para la generación de descriptores basados en propiedades termodinámicas y filogenéticas, expuestos en el capítulo 2, se considerarán la secuencia original de la proteína y serán generadas las variantes con respecto a la mutación reportada. De esta forma se creará un conjunto de datos basados en una variante y la respuesta asociada, lo cual corresponde a las diferencias de energía libre que provoca la sustitución del residuo.

Al conjunto de secuencias generado, se aplicará la codificación y digitalización de las propiedades fisicoquímicas, descritas en el punto anterior. La selección de las características se basa en un consenso con respecto a las incidencias de cada propiedad en cada secuencia.

Una vez se tenga el conjunto de espectros, modelos predictivos serán entrenados aplicando algoritmos de aprendizaje supervisado al set de datos de espectros de frecuencia. Se utilizarán las medidas de desempeño expuestas en el capítulo anterior. Los modelos serán validados mediante validación cruzada con un valor de $k = 10$, con el fin de evaluar el sobreajuste.

Los modelos serán comparados con los obtenidos en la fase de exploración de la metodología expuesta en el capítulo 2 y con los resultados finales a obtener. Esto con el fin de determinar, qué metodología o caracterización de datos, permite entregar un modelo con mejor desempeño o características deseables.

3.6.3 Identificación de residuos claves en espectros de frecuencia

3.6.4 Aplicación de técnicas de clustering, para categorización de espectros de frecuencia

Chapter 4

Filogenética, propiedades fisicoquímicas y minería de datos aplicados al diseño de mutaciones en secuencias de proteínas

Uno de los problemas de mayor relevancia para el campo de la ingeniería de proteínas, es el diseño inteligente de mutaciones, con el fin de obtener una propiedad fisicoquímica, mejorar las características o adicionar una nueva funcionalidad. Sin el hecho de incurrir en grandes costos económicos, de recursos humanos y computacionales.

En la sección 1.1, se mencionó que existen dos puntos de vista a la hora del diseño de mutaciones: Evolución dirigida y el diseño racional de proteínas. Sin embargo, ambas presentan el problema del poco espacio de exploración que pueden abarcar. A partir de ello, y con el fin de disminuir los tiempos experimentales y aumentar el espacio de búsqueda, métodos computacionales fueron implementados como herramientas de apoyo al diseño de mutantes.

A pesar de la existencia de dichos métodos, para aquellos basados en potenciales de energía y en simulaciones Monte Carlo, el tiempo computacional necesario para explorar la totalidad de mutaciones en una proteína en particular, escala a nivel cuadrático, siendo elevado y de limitado acceso para usuarios en general.

Una alternativa a los métodos computacionales basados en estrategias de potenciales de energía, son aquellos que emplean técnicas de minerías de datos y algoritmos de aprendizaje. Sin embargo, estos, principalmente se enfocan en el estudio de la estabilidad de la proteína ante sustituciones de aminoácidos. Por otro lado, los principales enfoques en cuanto a caracterización de residuos, se centran en propiedades termodinámicas y el ambiente, no siendo considerados conceptos filogenéticos ni propensión a cambios.

Tal como se expuso en el capítulo 3, la codificación de secuencias lineales puede ser realizada a través del uso de propiedades fisicoquímicas y posterior digitalización de éstas, aplicando transformadas de Fourier, esto último, permite caracterizar el ambiente y el aporte hacia las características que brindan los residuos, debido a las propiedades del espacio de frecuencias y el espectro como tal.

Apoyados en dicha codificación y centrados en la utilización de algoritmos de aprendizaje supervisado para el entrenamiento de modelos de clasificación o regresión, y utilizando la metodología expuesta a lo largo del capítulo 2, además del uso de herramientas computacionales como filtros de diseño. Se propone durante este capítulo el diseño e implementación de una herramienta computacional, basada en técnicas de minería de datos y aprendizaje de máquinas, que permita el diseño de mutaciones en variantes con características deseadas.

4.1 Hipótesis

En base al planteamiento del problema y a la motivación existente por el desarrollo de una herramienta computacional para el diseño de mutaciones, se plantea la siguiente hipótesis.

La digitalización de las propiedades fisicoquímicas por medio de transformadas de Fourier, en conjunto con algoritmos de aprendizaje supervisado, en combinación con herramientas computacionales para evaluar estabilidad y propensión, serán un enfoque suficiente para el desarrollo de una herramienta de diseño de mutaciones?

4.2 Objetivos

Dada la hipótesis planteada y en vista del desafío considerado, se exponen a continuación el objetivo general y los objetivos específicos.

4.2.1 Objetivo general

Diseñar, implementar, testear y depurar herramienta computacional para el diseño de mutaciones puntuales en variantes de proteínas, enfocada en el uso de minería de datos y aprendizaje supervisado y en la generación de estrategias de filtro de mutantes con respecto a puntos de vista termodinámicos y filogenéticos.

4.2.2 Objetivos específicos

Del objetivo general, nacen los siguientes objetivos específicos.

1. Implementar, evaluar y analizar, sistema de codificación de secuencias lineales por medio de propiedades fisicoquímicas y el uso de digitalización a partir de transformadas de Fourier.

2. Entrenar, evaluar y validar modelos basados en algoritmos de aprendizaje supervisado, con descriptores enfocados en espectros de frecuencias de propiedades fisicoquímicas.

3. Diseñar e implementar módulo de filtro de mutaciones por medio de propensión de la mutación y efecto en la estabilidad que provoque la sustitución.

4. Implementar y evaluar flujo de trabajo para nuevas mutaciones, mostrando el desempeño del modelo y los valores asociados a ésta, junto con algunas características de interés.

4.3 Métodología propuesta

Dada la hipótesis planteada y los objetivos propuestos, se diseña una metodología que contemple tanto la generación de la herramienta computacional, como el desarrollo de los modelos y su entrenamiento.

Un esquema representativo de los componentes principales de la metodología y cómo estos interactúan se expone en la Figura 4.1. El proceso general, se puede dividir en dos etapas: Generación de los modelos y Evaluación de nuevas mutaciones. En la primera etapa, se contempla un conjunto de datos, el cual es sometido a la etapa de digitalización, posterior a ello, se entrenan los modelos considerando como descriptores los espectros de frecuencia, para luego evaluar su desempeño. En la segunda etapa, nuevas mutaciones son propuestas y evaluadas en una etapa de filtro, aplicando criterios de estabilidad y propensión filogenética, para luego, someterse a los modelos predictivos generados y así evaluar si la mutación tendrá el efecto deseado o no.

Un punto importante a evaluar, consiste en el hecho que el conjunto inicial de datos debe ser lo suficientemente representativo en cuanto a la diversidad de ejemplos, es decir, si se evalúa la presencia o ausencia de una característica, no debe existir un desbalance entre ellas, ya que provocaría un sobreajuste en el modelo, tendiendo a predecir sobre la característica de mayor proporción.

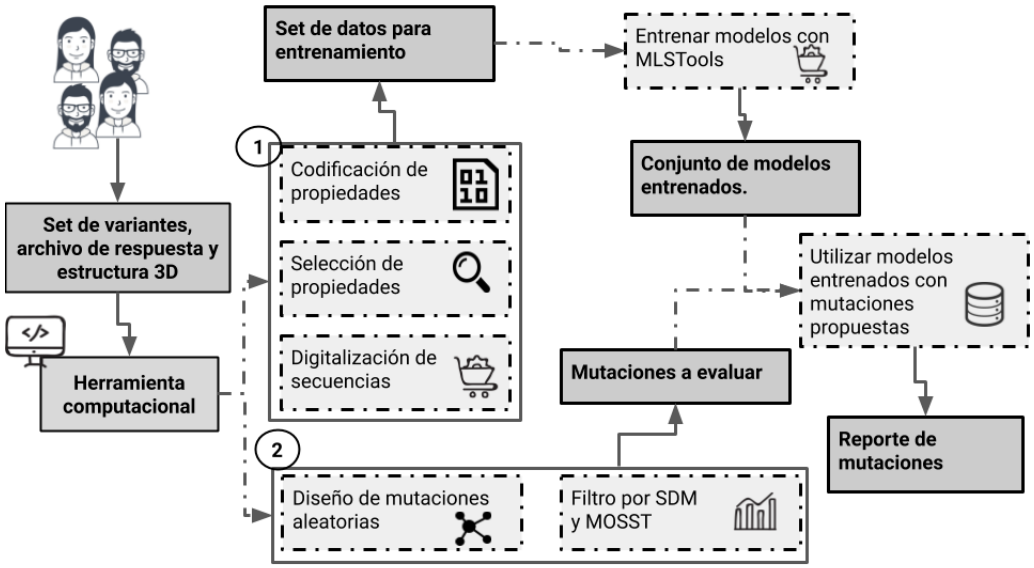


Fig. 4.1 Esquema representativo de la metodología propuesta para el diseño de mutaciones aplicando herramienta computacional a desarrollar

A continuación, se describe el conjunto de datos inicial, seguido de las etapas que componen la metodología y finalizando con la estrategia para el diseño e implementación de la herramienta computacional.

4.3.1 Conjunto de datos

El conjunto de datos corresponde a un grupo de variantes de una misma proteína, ordenados en un archivo en formato *.fasta[1], siendo la primera secuencia la proteína original, y el resto de secuencias las variantes con mutaciones reportadas. Además del conjunto de secuencias, es necesario un archivo en formato *.csv con la variable respuesta asociada a la variante, es decir, el valor de la propiedad fisicoquímica, característica, funcionalidad, etc., que provoca la mutación. Esto con el fin, de poder asignárselo a la secuencia y entrenar los correspondientes modelos.

Adicional a ambos inputs, se requiere de un archivo *.pdb el cual contenga la estructura de la proteína original, o en su defecto, el código PDB de la misma, esto debido a que la etapa de filtro, utiliza SDM como método de análisis de estabilidad de la proteína ante sustituciones de residuos, siendo un input para ésta, la estructura 3D.

[1]Formato de texto plano para la representación de secuencias.

4.3.2 Digitalización de secuencias lineales

A partir del conjunto de secuencias, los residuos son codificados aplicando las propiedades fisicoquímicas descritas en la base de datos AAindex [99], considerando la totalidad de propiedades descritas. Para ello, se implementarán scripts bajo el lenguaje de programación Python, los cuales tomen los residuos de cada secuencia y los transformen a un vector de tamaño n el cual corresponde a la cantidad de propiedades descritas en la base de datos. De esta manera, por cada secuencia se crea una matriz de tamaño $r \times n$ donde r representa la cantidad de residuos en la secuencia.

Dada esta matriz $r \times n$, se aplican técnicas de reducción de dimensionalidad y análisis de características, con el fin de seleccionar las propiedades fisicoquímicas más informativas y que generarían un aporte al entrenamiento de modelos. Debido a que son s secuencias existentes en el set de datos, se tendrán m propiedades fisicoquímicas por cada secuencia en el conjunto de elementos. La selección se basará en las propiedades consenso que abarquen la totalidad de las secuencias.

Es importante mencionar, que se espera que las propiedades informativas se mantengan a lo largo de la secuencia original y las variantes, ya que, cambios puntuales en los residuos, no alterarán de manera significativa la varianza que generen al conjunto de datos, alterando el orden de prioridad de las propiedades asociadas.

Una vez seleccionadas las propiedades fisicoquímicas a utilizar, se digitalizará su valor, con el fin de formar espectros de frecuencia asociados a dicho elemento; así, debido a que la selección previa, permite determinar un número p de propiedades informativas, por cada secuencia s en el conjunto de datos, se tendrán p espectros de frecuencias. Estos espectros se obtienen a partir del uso de Transformadas rápidas de Fourier (FFT por sus siglas en inglés) [29], para lograr dicha conversión, se implementarán scripts bajo lenguaje de programación Matlab, los cuales, reciban como entrada el conjunto de propiedades por cada secuencia y retornen los espectros de frecuencia por cada propiedad.

Finalmente, scripts basados en lenguaje de programación Python, completarán el conjunto de datos, considerando los espectros de frecuencia, seguidos de la respuesta que estos conllevan. De esta forma, generando el set de datos para el entrenamiento de modelos.

4.3.3 Entrenamiento de modelos

Los modelos se basarán en algoritmos de aprendizaje supervisado y se utilizará la misma metodología de exploración y selección de modelos basados en sus medidas de desempeño, expuestos en el capítulo 2. Se destaca que, el tipo de algoritmo a utilizar, depende de las características de la respuesta que se esté evaluando, es decir, si la respuesta presenta una

distribución continua, se utilizarán los métodos basados en regresión, en caso contrario, se utilizarán algoritmos basados en clasificación.

Con respecto al desbalance de clases, éste se evaluará de la misma forma expuesto en el capítulo 2, además, las respuestas del tipo continua serán analizadas mediante la evaluación de outliers y la tendencia a distribución normal que presente.

El desempeño de los modelos, será obtenido a partir de lo expuesto en el capítulo 2. Uno de los puntos importantes a considerar es el valor de estas medidas. Modelos de clasificación con medidas inferiores serán descartados y requerirán un nivel de análisis más detallado. Por otro lado, modelos de regresión con coeficientes de correlación inferiores a 0.6, también implica que deben ser analizados de la misma forma. De esta forma, se asegura un cierto nivel de confianza a la hora de aplicar los modelos. No obstante, dado a que a la metodología planteada en el capítulo 2 mejora las medidas de desempeño de los modelos iniciales, se espera que los casos expuestos de descarte no ocurran.

4.3.4 Diseño de mutaciones

Una vez los modelos se encuentren entrenados, será posible utilizarlos para evaluar nuevas mutaciones. Para ello, y con el fin de testear la herramienta, se implementará un script que permita mutar todas las posibilidades en cada residuo de la secuencia, es decir, para el residuo *i* se sustituirá 19 veces, en caso de que la mutación ya se encuentre reportada, ésta será descartada.

Una vez se tenga este conjunto de mutaciones, se aplicará un filtro basado en la propensión filogenética de dicha mutación y cómo afecta a la estabilidad el cambio propuesto. Para ello, se implementará servicios API (Application Programming Interface) que consuman las herramientas SDM [137] y MOSST [134], los cuales permitirán mencionar si la mutación es viable filogenéticamente y no provoca cambios en la estabilidad. Esto, generará una reducción de elementos en el conjunto de datos a estudiar, debido a que, no todas las mutaciones serán factibles.

Ya con las mutaciones factibles, se codificará las secuencias de las variantes utilizando el método descrito previamente y aplicando las propiedades fisicoquímicas seleccionadas, con el fin de obtener los espectros correspondientes.

Una vez obtenidos los espectros, se someten a los modelos entrenados y se obtiene la respuesta de interés, en base a las características del modelo. Se destaca que las respuestas serán reportadas en torno a intervalos de confianza, para métodos continuos, y, en forma de probabilidades, para el caso de respuestas categóricas.

4.3.5 Implementación herramienta computacional

La herramienta computacional será diseñada siguiendo el patrón de diseño Modelo-Vista-Controlador (MVC) [105] e implementada utilizando el paradigma de Programación Orientada a Objetos (POO) [180], además, debido a que el componente de la vista, será asociado a interfaz web, se utilizará la arquitectura Cliente-Servidor, con el fin de responder las solicitudes y ejecutar las acciones correspondientes. El uso de POO es debido a que permite una mayor estandarización de los módulos y facilita su re usabilidad, además de la comprensión y simpleza a la hora de programar dado a su cercanía con la realidad y al poder de abstracción que posee.

Con respecto al patrón de diseño, se expone a continuación, cuáles serán los principales elementos en cada componente, además de sus características y qué comprenden cada uno de estos.

Modelo

El modelo corresponde al conjunto de scripts y módulos que contienen toda la lógica de la herramienta y las funcionalidades principales, se hace alusión a que forma parte del "back-end" de la aplicación.

En este caso, se compondrá de los módulos de codificación, procesamiento de datos, entrenamiento de modelos, diseño de mutaciones y uso de servicios para la ejecución de MOSST y SDM, así como también los módulos de gestor de usuarios, notificaciones vía email, etc. Será implementado bajo el lenguaje de programación Python y utilizará algunas rutinas desarrolladas en Matlab. A su vez, cada módulo y sus componentes serán diseñados bajo el paradigma de Programación Orientada a Objetos y se utilizarán librerías externas como Pandas [126] para la manipulación del conjunto de datos, Numpy [173] y Scipy [133] para los análisis estadísticos y Scikit-Learn [140] para el entrenamiento de modelos.

Controlador

El controlador, hace referencia al gestor de las solicitudes de usuario desde la vista y recibe las respuestas de dichas solicitudes por parte del modelo, con el fin de ser expuestas al usuario.

Para este caso, el controlador se dividirá en dos componentes principales: Controlador de acciones en la vista y Controlador de respuestas en el modelo. El primero, será implementado en JavaScript y tendrá funcionalidades asociadas a jQuery, mientras que el segundo, será desarrollado bajo el lenguaje de programación Php.

La gran diferencia entre ambos, radica en el hecho de dónde se ejecutan, el primero, se encuentra principalmente condicionado por las acciones del usuario y su ejecución es en el navegador. Mientras que el segundo es una consecuencia del primero, es decir, una vez que se reciben las solicitudes, se establece la comunicación hacia el servidor por medio de Ajax (Asynchronous JavaScript And XML) y esto permite la ejecución del segundo, el cual, entrega una respuesta en formato JSON (JavaScript Object Notation), la que es capturada por el primero, y mostrada al usuario.

Vista

La vista, es el componente de visualización de la herramienta, es decir, es el componente en el cual el usuario puede interactuar, levantar solicitudes y exponer los resultados o respuestas que entregue el controlador y es conocido como "front-end".

Con el fin de poder representar las secciones principales que tendrá la herramienta, a continuación se exponen un conjunto de mockups (maqueta de diseño), en los cuales se exponen cuáles serán las principales funcionalidades y cómo se verán los resultados.

Generador de jobs

Los jobs, son los eventos generados asociados a un usuario y la carga de un conjunto de datos, los cuales deben ser codificados y entrenados los modelos, con el fin de evaluar las mutaciones posibles. Un esquema representativo del formulario asociado a la generación de jobs se aprecia en la Figura 4.2.

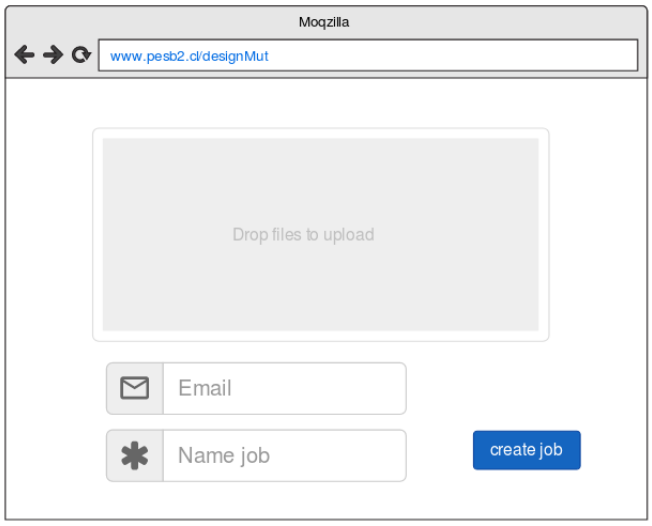


Fig. 4.2 Esquema representativo interfaz de creación de jobs.

En una primera instancia, el usuario deberá subir los archivos necesarios para entrenar los modelos, estos corresponden al conjunto de secuencias, el archivo de respuestas y la estructura 3D en formato *.pdb. Una vez suba los archivos, deberá ingresar el correo electrónico y el nombre del trabajo, esto para que él pueda identificarlo en el sistema. Cuando el usuario pulse el botón "create job", el sistema procesa la data y genera un ID asociado al identificador único del Job, posterior a ello, lo agrega al sistema de colas en el cual será procesado una vez le corresponda. Visualmente, el sistema retorna dicho ID, mostrándose en la interfaz y se notifica vía correo electrónico el estado del job, así como las notificaciones correspondientes del mismo.

Buscador de jobs

Adicional a la sección de crear jobs, la herramienta computacional permite buscar los trabajos generados, estos pueden encontrarse en 4 estados:

- **Creado**: se refiere al estado en el que el usuario sube la data y se ancla al sistema de colas.
- **En ejecución**: el job está siendo ejecutado por la herramienta computacional.
- **Finalizado**: el job fue ejecutado y es posible ver los resultados obtenidos.
- **Cancelado**: el job fue cancelado por el usuario.

Los jobs, pueden ser buscados en la interfaz del buscador de la herramienta web, mediante el ID o el nombre, ambos, son notificados vía correo electrónico, ya sea ante cambios de estado que éste sufra, o, al momento de crearse. Finalmente, una representación general del buscador, se observa en la Figura 4.3.

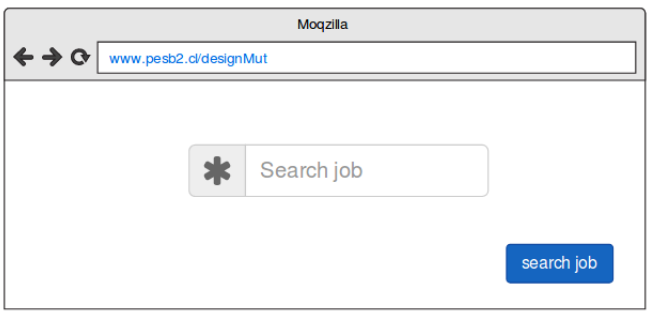


Fig. 4.3 Esquema representativo interfaz de búsqueda de jobs.

Visualizador de resultados

Una de las visualizaciones más relevantes, corresponde a los resultados involucrados en todo el proceso, estos, abarcan las propiedades fisicoquímicas estudiadas, los espectros de frecuencia y los residuos relevantes, los entrenamientos de los modelos y el listado de mutaciones favorables a evaluar. Un resumen general de los resultados esperados y la muestra de estos, es listada a continuación.

- **Descripción general**: Se genera una visualización del conjunto de secuencias y la distribución o frecuencia de la variable respuesta, se expone un resumen del conjunto de datos y se muestran patrones relacionados a alineamientos múltiples de secuencia. Además de una visualización de la proteína de interés, en su estructura 3D y las sustituciones que exhiben cada variante. Esto puede observarse en la Figura 4.4.

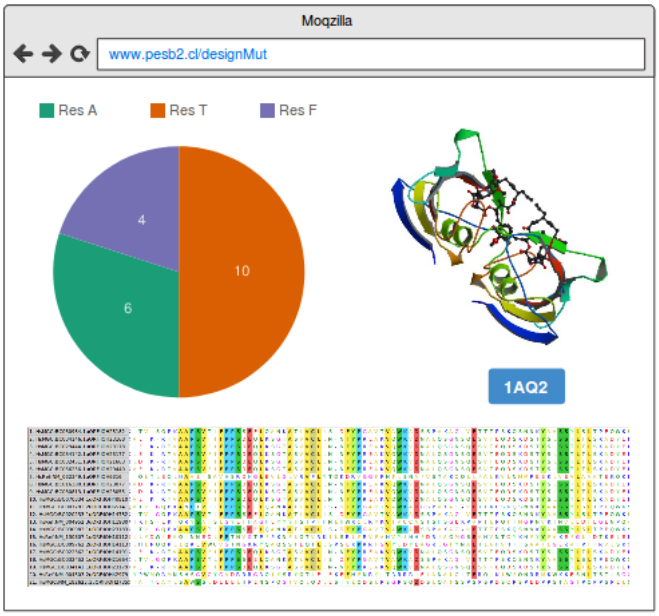


Fig. 4.4 Esquema representativo interfaz de descripción general del conjunto de datos.

- **Propiedades fisicoquímicas**: Una representación de las propiedades fisicoquímicas abarca cuáles fueron las seleccionadas y las definiciones de éstas, por otro lado se expone el aporte a la varianza que éstas presentan y cómo varían por cada secuencia. Un esquema representativo de diseño de la interfaz, puede observarse en la Figura 4.5.

- **Espectros de frecuencia**: Se exponen el conjunto de digitalizaciones de las propiedades fisicoquímicas, asociados a los residuos que brindan la mayor caracterización y cómo

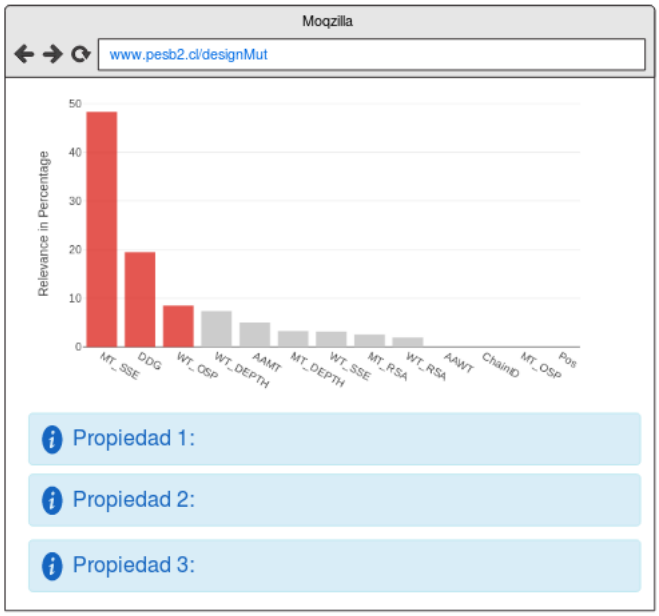


Fig. 4.5 Esquema representativo interfaz de visualización de propiedades fisicoquímicas.

estos inciden en el espectro. En la Figura 4.6, se expone un esquema de la visualización de los espectros, junto a los residuos relevantes.

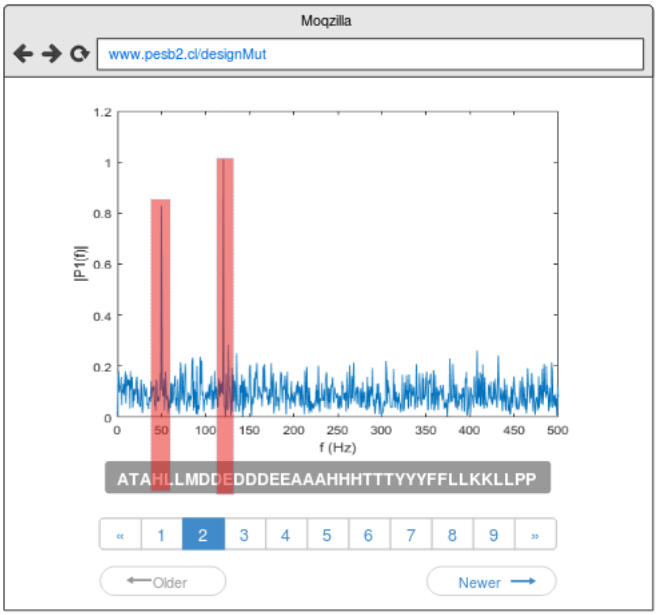


Fig. 4.6 Esquema representativo interfaz de visualización de espectros de frecuencias y residuos relevantes.

- **Entrenamiento de Modelos**: Los modelos se exponen asociados a las medidas de desempeño obtenidas y los miembros participantes en el conjunto de modelos. Esto es debido a que se utilizará la estrategia expuesta en el capítulo 2, por lo que se obtiene un sistema de meta-modelos, cuyo esquema representativo de visualización se observa en la Figura 4.7.

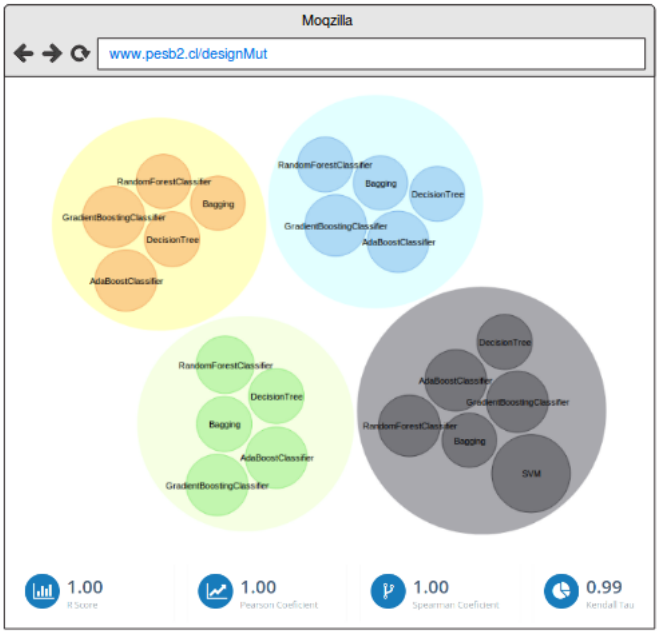


Fig. 4.7 Esquema representativo interfaz de visualización de los meta modelos y sus medidas de desempeño.

- **Mutaciones propuestas**: Para el conjunto de datos, se proponen diferentes mutaciones, evaluadas con respecto a la respuesta y factibilidad de éstas, a su vez, se exponen estadísticas asociadas a qué residuos son más factibles de mutar en base a las posiciones de estos.

4.3.6 Consideraciones generales

Como consideraciones generales, se tiene que el conjunto de secuencias o variantes a estudiar, debe poseer una respuesta reportada, es decir, la variable de interés debe conocerse de ante mano, con el fin de poder entrenar los modelos de clasificación o regresión y así poder analizar nuevas mutaciones.

Por otro lado, la secuencia original, debe presentar su estructura en formato *.pdb, ya que, se requiere para el uso de SDM, la cual es la herramienta de filtro asociada a la estabilidad de la proteína con respecto a los cambios o sustituciones propuestas.

Un punto importante a destacar, es que, todos los pasos relacionados a la generación de modelos, implicando desde la codificación y posterior digitalización de las secuencias hasta la fase de evaluación del desempeño, serán ejecutadas en torno a Jobs que cree el usuario y serán procesadas internamente en servidor en un gestor de colas, con el fin de optimizar los procesos de ejecución. Esto es debido, a que el tiempo de entrenamiento puede ser elevado si existen muchas variantes en el conjunto de datos, razón por la cual, una vez que el job finalice, se notificará vía email al usuario, en donde, él podrá acceder al sistema y revisar los resultados obtenidos.

Se propone esta herramienta como un servicio de exploración, en el cual, se evalúan un conjunto de mutaciones y se sugieren nuevos elementos, de tal manera, que el usuario pueda tener un poco más claro el panorama. No obstante, las mutaciones son proposiciones basados en los filtros y en los modelos aplicados. Por lo que, está condicionado por el conjunto de datos inicial y las medidas de desempeño obtenidas.

Chapter 5

Planificación y estado de avance.

Es posible pensar en este proyecto de Tesis, como un conjunto de postulaciones y planteamientos de nuevas metodologías basadas en el uso de técnicas de minería de datos que permitan estudiar mutaciones en conjuntos de datos y que culminan con una gran temática, que aborda la implementación de una herramienta computacional para el diseño de mutaciones.

Dado esto, el proyecto en sí, puede dividirse en tres grandes objetivos, de los cuales, los conocimientos y destrezas adquiridas en los primeros dos, son necesarias para cumplir con el tercer gran objetivo. Esto, es posible observarlo en la Figura 5.1.

Fig. 5.1 Esquema representativo de objetivos generales involucrados en el desarrollo del proyecto.

El primer objetivo, se basa en la construcción de modelos de clasificación o regresión basados en algoritmos de aprendizaje supervisado, enfocados en el estudio de mutaciones

puntuales en una proteína y cómo afectan éstas en términos de una respuesta conocida, como por ejemplo, estabilidad, productividad, actividad, etc. Esto, va en directo beneficio del estudio de nuevas mutantes o variantes en proteínas de interés, apoyados en una metodología que maximiza el desempeño de los modelos y sin incurrir en costos computacionales elevados. No obstante, el hecho de manipular nuevas mutaciones, requiere una verificación experimental. Sin embargo, esto permite minimizar los costos económicos y de recursos humanos que conlleva estudiar un gran conjunto de espacios muestrales de mutaciones.

Por otro lado, el segundo objetivo se basa principalmente en representar secuencias lineales de proteínas a partir de la digitalización de propiedades fisicoquímicas y empleando transformadas de Fourier como representación de espectros de frecuencias. Si bien, el enfoque principal es la representación en sí, el objetivo también abarca el cómo utilizar estas representaciones para el entrenamiento de modelos de clasificación/regresión o el reconocimiento de patrones e identificación de residuos que aportan a las propiedades fisicoquímicas.

Finalmente, el tercer gran objetivo, comprende el desarrollo de una herramienta computacional, basada en técnicas de minería de datos para el diseño de mutaciones en proteínas de interés. El cómo se abarcará esta problemática, comprende, por un lado, definir representaciones de las secuencias lineales y cómo codificar sus propiedades fisicoquímicas, en conjunto, con el entrenamiento de modelos de clasificación/regresión que permitan evaluar nuevas mutantes. Es decir, los enfoques propuestos en los capítulos 3 y 2, respectivamente. Demostrando así, las relaciones existentes entre cada capítulo y cómo estos apuntan a un objetivo general que conlleva aplicar minería de datos en el estudio de mutaciones puntuales asociados a la ingeniería de proteínas.

Con el fin de trazar los panoramas asociados al desarrollo de cada metodología y exponer el estado de avance del proyecto en cuanto a las diferentes actividades desarrolladas, en el presente capítulo, se expone un conjunto de actividades resumen de cada objetivo, asociados a un tiempo estimativo que conlleva el cumplimiento de estos y a su vez, las tareas ya han sido desarrolladas, junto con las actividades pendientes a realizar.

5.1 Planificación

La planificación se centra en cumplir los tres grandes objetivos y se plantea una estimación del tiempo que conlleva el desarrollo de cada una de las temáticas. Se exponen diferentes tareas generales que aportan a cumplir los objetivos y el tiempo estimativo, considerando un desarrollo general de 3 años como máximo para todas las actividades planteadas. Es

importante mencionar y recalcar que este itinerario es un estimativo y sólo se exponen las tareas principales para el desarrollo del proyecto.

En las Tablas 5.1, 5.2 y 5.3 se exponen las actividades relacionadas al objetivo principal que aportan a cumplir, además del tiempo estimativo de desarrollo.

Resumen de actividades enfocadas en el desarrollo de modelos de clasificación		
#	**Actividad a desarrollar**	**Tiempo (semanas)**
1.	Búsqueda de mutaciones en bases de datos y reportes publicados	3
2.	Preparación y análisis del conjunto de datos a estudiar	3
3.	Describir set de datos enfocados en propiedades termodinámicas	5
4.	Describir set de datos enfocados en conceptos filogenéticos	5
5.	Implementar algoritmos de aprendizaje supervisado	3
6.	Implementar metodología propuesta	2
7.	Entrenar modelos de clasificación/regresión para conjuntos de proteínas seleccionados	5
8.	Revisar resultados y reportar conclusiones	5
9.	Implementar herramienta computacional asociada	5
10.	Testear y generar depuración de herramienta	3
	Total Semanas	**39**

Table 5.1 Resumen de actividades principales a desarrollar, enfocadas el desarrollo de modelos predictivos para mutaciones puntuales en proteínas.

Resumen de actividades enfocadas en la codificación de secuencias lineales		
#	**Actividad a desarrollar**	**Tiempo (semanas)**
1.	Búsqueda de secuencias lineales	2
2.	Búsqueda de conjuntos de datos reportados con respuesta conocida	2
3.	Codificación de secuencias lineales empleando propiedades descritas en AAindex	3
4.	Selección de propiedades informativas y generación de propiedades consenso	4
5.	Digitalización de propiedades fisicoquímicas empleando transformadas de Fourier	4
6.	Identificación de residuos relevantes para la propiedad fisicoquímica	3
7.	Aplicación de métodos de clustering para búsqueda de patrones en espectros de frecuencia	4
8.	Entrenamiento de modelos de clasificación/regresión con descriptores basados en espectros de frecuencia	4

9.	Revisar resultados y reportar conclusiones	5
10.	Implementar herramienta computacional asociada	5
11.	Testear y generar depuración de herramienta	3
	Total Semanas	**39**

Table 5.2 Resumen de actividades asociadas a la codificación y digitalización de propiedades fisicoquímicas

Resumen de actividades enfocadas en implementación de herramienta para diseño de mutaciones		
#	**Actividad a desarrollar**	**Tiempo (semanas)**
1.	Implementación módulo codificación de secuencias lineales	3
2.	Implementación módulo selección de propiedades	3
3.	Implementación módulo digitalización de propiedades	3
4.	Implementación módulo entrenamiento de modelos	4
5.	Implementación servicio API MOSST	4
6.	Implementación servicio API SDM	4
7.	Implementación servicio propuesta de mutaciones	5
8.	Implementación servicio gestor de sistema de colas	4
9.	Implementación controlador de solicitudes	7
10.	Implementación de servicio creación de jobs en Front-end	5
11.	Implementación de servicio búsqueda de jobs	2
12.	Implementación de servicio visualización de resultados	10
13.	Implementación visualización de mutaciones propuestas	6
14.	Implementación servicio de notificaciones	2
15.	Teste y debug herramienta computacional	5
	Total Semanas	**79**

Table 5.3 Resumen de actividades asociadas al desarrollo de herramienta computacional web para diseño de mutaciones

Para el desarrollo del primer objetivo, se contempla un total de 39 semanas, de la misma forma que para el objetivo 2, mientras que para el objetivo 3, se contempla un total de 79 semanas. En total se estima un total de 3 años, para poder cumplir con los objetivos planteados, tiempo en el cual, se piensa desarrollar las actividades propuestas previamente.

Un punto importante a destacar, es que en sí, cada objetivo consta del desarrollo de una herramienta en particular, enfocada en el objetivo en sí, con el enfoque principal de brindar un apoyo sustancial a investigadores asociados al estudio de mutaciones e involucrados en ingeniería de proteínas.

5.2 Estado de avance

Con respecto al estado de avance, actualmente se tiene desarrollado un porcentaje significativo de lo planteado en el capítulo 1, al menos, en cuanto a desarrollo de módulos, implementación de código fuente y búsqueda de set de datos y conjuntos de proteínas a evaluar.

A modo resumen, se expone una variación de la Tabla 5.1, adicionando el avance en el que se encuentra dicho desarrollo en términos de porcentaje, la cual se observa en la Tabla 5.4.

Resumen de actividades enfocadas en el desarrollo de modelos de clasificación			
#	**Actividad a desarrollar**	**Tiempo (semanas)**	**Estado Avance**
1.	Búsqueda de mutaciones en bases de datos y reportes publicados	3	100%
2.	Preparación y análisis del conjunto de datos a estudiar	3	100%
3.	Describir set de datos enfocados en propiedades termodinámicas	5	100%
4.	Describir set de datos enfocados en conceptos filogenéticos	5	0%
5.	Implementar algoritmos de aprendizaje supervisado	3	100%
6.	Implementar metodología propuesta	2	50%
7.	Entrenar modelos de clasificación/regresión para conjuntos de proteínas seleccionados	5	0%
8.	Revisar resultados y reportar conclusiones	5	0%
9.	Implementar herramienta computacional asociada	5	0%
10.	Testear y generar depuración de herramienta	3	0%
	Total Semanas	**39**	**38%**

Table 5.4 Resumen y estado de actividades principales asociadas al desarrollo de modelos de clasificación en mutaciones puntuales.

Se puede apreciar, que se ha completado cerca del 40% del trabajo asociado al desarrollo de modelos de predictivos, siendo tareas pendientes, caracterizar mutaciones desde puntos

de vista filogenéticos, el entrenamiento de modelos y el desarrollo de una herramienta computacional asociada al entrenamiento de conjuntos de datos para predecir respuestas conocidas.

Finalmente, tanto la planificación como el estado de avance expuesto, se muestran con el fin de denotar la factibilidad del proyecto y cuáles son las actividades que se tienen consideradas en cada etapa de desarrollo. Como se mencionó previamente, el trabajo de tesis, se plantea como un conjunto de metodologías e implementación de herramientas computacionales, que permitan el estudio de las mutaciones y sirva de apoyo en el área de ingeniería de proteínas, culminando con el ambicioso objetivo de generar una herramienta que permita el diseño de mutantes, siendo ésta la gran premisa de este proyecto de tesis.

Referencias

[1] Abdelaziz, A., Elhoseny, M., Salama, A. S., and Riad, A. (2018). A machine learning model for improving healthcare services on cloud computing environment. *Measurement*, 119:117 – 128.

[2] Abola, E. E., Bernstein, F. C., and Koetzle, T. F. (1984). The protein data bank. In *Neutrons in Biology*, pages 441–441. Springer.

[3] Alexov, E., Zhang, J., Wang, L., Zhenirovskyy, M., Gao, Y., and Zhang, Z. (2012). Predicting folding free energy changes upon single point mutations. *Bioinformatics*, 28(5):664–671.

[4] Alm, C. O., Roth, D., and Sproat, R. (2005). Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. Association for Computational Linguistics.

[5] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

[6] Amari, S.-i. and Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789.

[7] Ancien, F., Pucci, F., Godfroid, M., and Rooman, M. (2018). Prediction and interpretation of deleterious coding variants in terms of protein structural stability. *Scientific reports*, 8(1):4480.

[8] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989.

[9] Arel, I., Rose, D. C., Karnowski, T. P., et al. (2010). Deep machine learning-a new frontier in artificial intelligence research. *IEEE computational intelligence magazine*, 5(4):13–18.

[10] Arnold, F. H. (1998). Design by directed evolution. *Accounts of chemical research*, 31(3):125–131.

[11] Artac, M., Jogan, M., and Leonardis, A. (2002). Incremental pca for on-line visual learning and recognition. In *Object recognition supported by user interaction for service robots*, volume 3, pages 781–784. IEEE.

[12] Barandiaran, I. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8).

[13] Barenboim, M., Masso, M., Vaisman, I. I., and Jamison, D. C. (2008). Statistical geometry based prediction of nonsynonymous snp functional effects using random forest and neuro-fuzzy classifiers. *Proteins: Structure, Function, and Bioinformatics*, 71(4):1930–1939.

[14] Battiti, R. (1992). First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166.

[15] Bedbrook, C. N., Yang, K. K., Robinson, J. E., Gradinaru, V., and Arnold, F. H. (2019). Machine learning-guided channelrhodopsin engineering enables minimally-invasive optogenetics.

[16] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

[17] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

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

[19] Berry, M. J. and Linoff, G. S. (2004). *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons.

[20] Bhargava, N., Sharma, G., Bhargava, R., and Mathuria, M. (2013). Decision tree analysis on j48 algorithm for data mining. *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6).

[21] Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'donovan, C., Phan, I., et al. (2003). The swiss-prot protein knowledgebase and its supplement trembl in 2003. *Nucleic acids research*, 31(1):365–370.

[Bordner and Abagyan] Bordner, A. J. and Abagyan, R. A. Large-scale prediction of protein geometry and stability changes for arbitrary single point mutations. *Proteins: Structure, Function, and Bioinformatics*, 57(2):400–413.

[23] Braha, D. and Shmilovici, A. (2002). Data mining for improving a cleaning process in the semiconductor industry. *IEEE Transactions on Semiconductor Manufacturing*, 15(1):91–101.

[24] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

[25] Breiman, L. (1998). Arcing classifier (with discussion and a rejoinder by the author). *Ann. Statist.*, 26(3):801–849.

[26] Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine learning*, 36(1-2):85–103.

[27] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[28] Breiman, L. (2017). *Classification and regression trees*. Routledge.

[29] Brigham, E. O. and Brigham, E. O. (1988). *The fast Fourier transform and its applications*, volume 448. prentice Hall Englewood Cliffs, NJ.

[30] Broom, A., Jacobi, Z., Trainor, K., and Meiering, E. M. (2017a). Computational tools help improve protein stability but with a solubility tradeoff. *J Biol Chem*, 292(35):14349–14361. 28710274[pmid].

[31] Broom, A., Jacobi, Z., Trainor, K., and Meiering, E. M. (2017b). Computational tools help improve protein stability but with a solubility tradeoff. *Journal of Biological Chemistry*, 292(35):14349–14361.

[32] Brownlee, J. (2017). Why one-hot encode data in machine learning.

[33] Cadet, F., Fontaine, N., Vetrivel, I., Chong, M. N. F., Savriama, O., Cadet, X., and Charton, P. (2018). Application of fourier transform and proteochemometrics principles to protein engineering. *BMC bioinformatics*, 19(1):382.

[34] Canutescu, A. A., Shelenkov, A. A., and Dunbrack Jr, R. L. (2003). A graph-theory algorithm for rapid protein side-chain prediction. *Protein science*, 12(9):2001–2014.

[35] CAO, Y., MIAO, Q.-G., LIU, J.-C., and GAO, L. (2013). Advance and prospects of adaboost algorithm. *Acta Automatica Sinica*, 39(6):745 – 758.

[36] Capriotti, E., Fariselli, P., and Casadio, R. (2005a). I-mutant2. 0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic acids research*, 33(suppl_2):W306–W310.

[37] Capriotti, E., Fariselli, P., and Casadio, R. (2005b). I-mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res*, 33(Web Server issue):W306–W310. 15980478[pmid].

[38] Capriotti, E., Fariselli, P., Rossi, I., and Casadio, R. (2008a). A three-state prediction of single point mutations on protein stability changes. *BMC bioinformatics*, 9(2):S6.

[39] Capriotti, E., Fariselli, P., Rossi, I., and Casadio, R. (2008b). A three-state prediction of single point mutations on protein stability changes. *BMC Bioinformatics*, 9(2):S6.

[40] Carpenter, J. F., Pikal, M. J., Chang, B. S., and Randolph, T. W. (1997). Rational design of stable lyophilized protein formulations: some practical advice. *Pharmaceutical research*, 14(8):969–975.

[41] Case, D. A., Cheatham III, T. E., Darden, T., Gohlke, H., Luo, R., Merz Jr, K. M., Onufriev, A., Simmerling, C., Wang, B., and Woods, R. J. (2005). The amber biomolecular simulation programs. *Journal of computational chemistry*, 26(16):1668–1688.

[42] Castillo, I., Schmidt-Hieber, J., Van der Vaart, A., et al. (2015). Bayesian linear regression with sparse priors. *The Annals of Statistics*, 43(5):1986–2018.

[43] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

[44] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[45] Chen, C., Lu, Z., and Ciucci, F. (2017). Data mining of molecular dynamics data reveals li diffusion characteristics in garnet li7la3zr2o12. *Scientific Reports*, 7:40769 EP –. Article.

[46] Chen, J., Huang, H., Tian, S., and Qu, Y. (2009). Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3, Part 1):5432–5435.

[47] Chen, M., Hao, Y., Hwang, K., Wang, L., and Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, 5:8869–8879.

[48] Chien, C.-F. and Chen, L.-F. (2008). Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with Applications*, 34(1):280–290.

[49] Cohen, P., West, S. G., and Aiken, L. S. (2014). *Applied multiple regression/correlation analysis for the behavioral sciences*. Psychology Press.

[50] Cooley, R., Mobasher, B., Srivastava, J., et al. (1997). Web mining: Information and pattern discovery on the world wide web. In *ictai*, volume 97, pages 558–567.

[51] Cosic, I., Cosic, D., and Lazar, K. (2016). Analysis of tumor necrosis factor function using the resonant recognition model. *Cell biochemistry and biophysics*, 74(2):175–180.

[52] Curtarolo, S., Morgan, D., Persson, K., Rodgers, J., and Ceder, G. (2003). Predicting crystal structures with data mining of quantum calculations. *Phys. Rev. Lett.*, 91:135503.

[53] Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227 – 248.

[54] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387.

[55] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[56] Duan, L., Street, W. N., and Xu, E. (2011). Healthcare information systems: data mining methods in the creation of a clinical recommender system. *Enterprise Information Systems*, 5(2):169–181.

[57] Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327.

[58] Dunham, M. H. (2006). *Data mining: Introductory and advanced topics*. Pearson Education India.

[59] Duygulu, P., Barnard, K., de Freitas, J. F. G., and Forsyth, D. A. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Computer Vision — ECCV 2002*, pages 97–112, Berlin, Heidelberg. Springer Berlin Heidelberg.

[60] Eswar, N., Webb, B., Marti-Renom, M. A., Madhusudhan, M., Eramian, D., Shen, M.-y., Pieper, U., and Sali, A. (2006). Comparative protein structure modeling using modeller. *Current protocols in bioinformatics*, 15(1):5–6.

[61] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37.

[62] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996b). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

[63] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. (1996c). Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88.

[64] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

[65] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

[66] Forbes, S. A., Bindal, N., Bamford, S., Cole, C., Kok, C. Y., Beare, D., Jia, M., Shepherd, R., Leung, K., Menzies, A., Teague, J. W., Campbell, P. J., Stratton, M. R., and Futreal, P. A. (2010). Cosmic: mining complete cancer genomes in the catalogue of somatic mutations in cancer. *Nucleic Acids Research*, 39(suppl_1):D945–D950.

[67] Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133.

[68] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

[69] Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics AND Data Analysis*, 38(4):367 – 378. Nonlinear Methods and Data Mining.

[70] Gallou, C., Junien, C., Joly, D., Staroz, F., Orfanelli, M. T., and Béroud, C. (1998). Software and database for the analysis of mutations in the VHL gene. *Nucleic Acids Research*, 26(1):256–258.

[71] Ge, Z., Song, Z., Ding, S. X., and Huang, B. (2017). Data mining and analytics in the process industry: The role of machine learning. *IEEE Access*, 5:20590–20616.

[72] Getov, I., Petukh, M., and Alexov, E. (2016a). Saafec: Predicting the effect of single point mutations on protein folding free energy using a knowledge-modified mm/pbsa approach. *Int J Mol Sci*, 17(4):512–512. 27070572[pmid].

[73] Getov, I., Petukh, M., and Alexov, E. (2016b). Saafec: predicting the effect of single point mutations on protein folding free energy using a knowledge-modified mm/pbsa approach. *International journal of molecular sciences*, 17(4):512.

[74] Golub, G. H., Heath, M., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223.

[75] Golub, G. H. and Van Loan, C. F. (1980). An analysis of the total least squares problem. *SIAM journal on numerical analysis*, 17(6):883–893.

[76] Gossage, L., Pires, D., Olivera-Nappa, A., A. Asenjo, J., Bycroft, M., Blundell, T., and Eisen, T. (2014). An integrated computational approach can classify vhl missense mutations according to risk of clear cell renal carcinoma. *Human molecular genetics*, 23.

[77] Granitto, P. M., Furlanello, C., Biasioli, F., and Gasperi, F. (2006). Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems*, 83(2):83–90.

[78] Graybill, F. A. (1976). *Theory and application of the linear model*. Number QA279. G72 1976. Duxbury press North Scituate, MA.

[79] Guex, N. and Peitsch, M. C. (1997). Swiss-model and the swiss-pdb viewer: An environment for comparative protein modeling. *ELECTROPHORESIS*, 18(15):2714–2723.

[80] Guyon, I., Boser, B., and Vapnik, V. (1993). Automatic capacity tuning of very large vc-dimension classifiers. In *Advances in neural information processing systems*, pages 147–155.

[81] Han, J. and Gao, J. (2009). Research challenges for data mining in science and engineering. *Next Generation of Data Mining*, pages 1–18.

[82] Hand, D. J. (2006). Data mining. *Encyclopedia of Environmetrics*, 2.

[83] Hans, C. (2009). Bayesian lasso regression. *Biometrika*, 96(4):835–845.

[84] Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

[85] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[86] HECHT-NIELSEN, R. (1992). Iii.3 - theory of the backpropagation neural network**based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee. In Wechsler, H., editor, *Neural Networks for Perception*, pages 65 – 93. Academic Press.

[87] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[88] Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001). Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer.

[89] Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

[90] Hofmann, M. and Klinkenberg, R. (2013). *RapidMiner: Data mining use cases and business analytics applications*. CRC Press.

[91] Hssina, B., Merbouha, A., Ezzikouri, H., and Erritali, M. (2014). A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2):0–0.

[92] Hua, S. and Sun, Z. (2001). A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of molecular biology*, 308(2):397–407.

[93] Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM.

[94] Jain, A. K., Dubes, R. C., et al. (1988). *Algorithms for clustering data*, volume 6. Prentice hall Englewood Cliffs.

[95] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

[96] Jespersen, M. C., Peters, B., Nielsen, M., and Marcatili, P. (2017). Bepipred-2.0: improving sequence-based b-cell epitope prediction using conformational epitopes. *Nucleic acids research*, 45(W1):W24–W29.

[97] John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.

[98] Jolliffe, I. (2011). *Principal component analysis*. Springer.

[99] Kawashima, S. and Kanehisa, M. (2000). Aaindex: Amino acid index database. *Nucleic Acids Research*, 28(1):374–374.

[100] Keller, J. M., Gray, M. R., and Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(4):580–585.

[101] Kent, W. J. (2002). Blat—the blast-like alignment tool. *Genome research*, 12(4):656–664.

[102] Khan, S. and Vihinen, M. (2010). Performance of protein stability predictors. *Human Mutation*, 31(6):675–684.

[103] Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

[104] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24.

[105] Krasner, G. E., Pope, S. T., et al. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49.

[106] Larrañaga, P., Calvo, B., Santana, R., Bielza, C., Galdiano, J., Inza, I., Lozano, J. A., Armañanzas, R., Santafé, G., Pérez, A., and Robles, V. (2006). Machine learning in bioinformatics. *Briefings in Bioinformatics*, 7(1):86–112.

[107] Leaver-Fay, A., Tyka, M., Lewis, S. M., Lange, O. F., Thompson, J., Jacak, R., Kaufman, K. W., Renfrew, P. D., Smith, C. A., Sheffler, W., et al. (2011). Rosetta3: an object-oriented software suite for the simulation and design of macromolecules. In *Methods in enzymology*, volume 487, pages 545–574. Elsevier.

[108] Lee, J. K., Williams, P. D., and Cheon, S. (2008). Data mining in genomics. *Clinics in Laboratory Medicine*, 28(1):145–166.

[109] Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In Nédellec, C. and Rouveirol, C., editors, *Machine Learning: ECML-98*, pages 4–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

[110] Li, M., Wang, J., and Chen, J. (2008). A fast agglomerate algorithm for mining functional modules in protein interaction networks. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 1, pages 3–7.

[111] Li, M., Wang, J., and Chen, J. (2008). A fast agglomerate algorithm for mining functional modules in protein interaction networks. In *2008 International conference on biomedical engineering and informatics*, volume 1, pages 3–7. IEEE.

[112] Liao, H. and Xu, Z. (2015). Approaches to manage hesitant fuzzy linguistic information based on the cosine distance and similarity measures for hfltss and their application in qualitative decision making. *Expert Systems with Applications*, 42(12):5328 – 5336.

[113] Liszewski, K. (2015). Speeding up the protein assembly line. *Genetic Engineering & Biotechnology News*, 35(04):1–10.

[114] Louppe, G. and Geurts, P. (2012). Ensembles on random patches. In Flach, P. A., De Bie, T., and Cristianini, N., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 346–361, Berlin, Heidelberg. Springer Berlin Heidelberg.

[115] Lutz, S. (2010). Beyond directed evolution—semi-rational protein engineering and design. *Current opinion in biotechnology*, 21(6):734–743.

[116] Lutz, S., Bornscheuer, U. T., et al. (2009). *Protein engineering handbook*, volume 1. Wiley Online Library.

[117] Lyskov, S. and Gray, J. J. (2008). The rosettadock server for local protein–protein docking. *Nucleic acids research*, 36(suppl_2):W233–W238.

[118] Ma, X., Wu, Y.-J., Wang, Y., Chen, F., and Liu, J. (2013). Mining smart card data for transit riders' travel patterns. *Transportation Research Part C: Emerging Technologies*, 36:1–12.

[119] Maesschalck, R. D., Jouan-Rimbaud, D., and Massart, D. (2000). The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18.

[120] Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

[121] Mao, L., Wang, Y., Liu, Y., and Hu, X. (2004). Molecular determinants for atp-binding in proteins: A data mining and quantum chemical analysis. *Journal of Molecular Biology*, 336(3):787 – 807.

[122] Martin, A. J., Contreras-Riquelme, S., Dominguez, C., and Perez-Acle, T. (2017). Loto: a graphlet based method for the comparison of local topology between gene regulatory networks. *PeerJ*, 5:e3052.

[123] Masso, M. and Vaisman, I. I. (2008). Accurate prediction of stability changes in protein mutants by combining machine learning with structure based computational mutagenesis. *Bioinformatics*, 24(18):2002–2009.

[124] Masso, M. and Vaisman, I. I. (2010). Auto-mute: web-based tools for predicting stability changes in proteins due to single amino acid replacements. *Protein Engineering, Design and Selection*, 23(8):683–687.

[125] McGuffin, L. J., Atkins, J. D., Salehe, B. R., Shuid, A. N., and Roche, D. B. (2015). Intfold: an integrated server for modelling protein structures and functions from amino acid sequences. *Nucleic acids research*, 43(W1):W169–W173.

[126] McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

[127] Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA.

[128] Michie, D., Spiegelhalter, D. J., Taylor, C., et al. (1994). Machine learning. *Neural and Statistical Classification*, 13.

[129] Muggleton, S., King, R. D., and Stenberg, M. J. E. (1992). Protein secondary structure prediction using logic-based machine learning. *Protein Engineering, Design and Selection*, 5(7):647–657.

[130] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

[131] Obenshain, M. K. (2004). Application of data mining techniques to healthcare data. *Infection Control &; Hospital Epidemiology*, 25(8):690–695.

[132] Odorico, M. and Pellequer, J.-L. (2003). Bepitope: predicting the location of continuous epitopes and patterns in proteins. *Journal of Molecular Recognition*, 16(1):20–22.

[133] Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20.

[134] Olivera-Nappa, A., Andrews, B. A., and Asenjo, J. A. (2011). Mutagenesis objective search and selection tool (mosst): an algorithm to predict structure-function related mutations in proteins. *BMC Bioinformatics*, 12(1):122.

[135] Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109.

[136] Ozbudak, O. and Dokur, Z. (2014). Protein fold classification using kohonen's self-organizing map. In *IWBBIO*, pages 903–911.

[137] Pandurangan, A. P., Ochoa-Montaño, B., Ascher, D. B., and Blundell, T. L. (2017). Sdm: a server for predicting effects of mutations on protein stability. *Nucleic Acids Res*, 45(W1):W229–W235. 28525590[pmid].

[138] Parthiban, V., Gromiha, M. M., and Schomburg, D. (2006). Cupsat: prediction of protein stability upon point mutations. *Nucleic Acids Res*, 34(Web Server issue):W239–W242. 16845001[pmid].

[139] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE.

[140] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[141] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1226–1238.

[142] Perlibakas, V. (2004). Distance measures for pca-based face recognition. *Pattern Recognition Letters*, 25(6):711 – 724.

[143] Petukh, M., Dai, L., and Alexov, E. (2016). Saambe: webserver to predict the charge of binding free energy caused by amino acids mutations. *International journal of molecular sciences*, 17(4):547.

[144] Phillips, J. C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., Chipot, C., Skeel, R. D., Kale, L., and Schulten, K. (2005). Scalable molecular dynamics with namd. *Journal of computational chemistry*, 26(16):1781–1802.

[145] Potapov, V., Cohen, M., and Schreiber, G. (2009). Assessing computational methods for predicting protein stability upon mutation: good on average but not in the details. *Protein Engineering, Design and Selection*, 22(9):553–560.

[146] Quan, L., Lv, Q., and Zhang, Y. (2016a). Strum: structure-based prediction of protein stability changes upon single-point mutation. *Bioinformatics*, 32(19):2936–2946. 27318206[pmid].

[147] Quan, L., Lv, Q., and Zhang, Y. (2016b). Strum: structure-based prediction of protein stability changes upon single-point mutation. *Bioinformatics*, 32(19):2936–2946.

[148] Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. (2015). Big data meets quantum chemistry approximations: The d-machine learning approach. *Journal of Chemical Theory and Computation*, 11(5):2087–2096.

[149] Rebhan, M., Chalifa-Caspi, V., Prilusky, J., and Lancet, D. (1998). Genecards: a novel functional genomics compendium with automated data mining and query reformulation support. *Bioinformatics*, 14(8):656–664.

[150] Reetz, M. T., Kahakeaw, D., and Lohmer, R. (2008). Addressing the numbers problem in directed evolution. *ChemBioChem*, 9(11):1797–1804.

[151] Release, S. (2016). 1: Maestro. *Schrödinger, LLC, New York, NY, USA*.

[152] Rohl, C. A., Strauss, C. E., Misura, K. M., and Baker, D. (2004). Protein structure prediction using rosetta. In *Methods in enzymology*, volume 383, pages 66–93. Elsevier.

[153] Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., and Suter, B. W. (1990). The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298.

[154] Saeys, Y., Abeel, T., and Van de Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 313–325. Springer.

[155] Saha, S. and Raghava, G. (2008). Abcpred benchmarking datasets. 2006a.

[156] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science.

[157] Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München.

[158] Schmidhuber, J. (1995). On learning how to learn learning strategies.

[159] Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

[160] Schueler-Furman, O. and Baker, D. (2003). Conserved residue clustering and protein structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 52(2):225–235.

[161] Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F., and Serrano, L. (2005). The foldx web server: an online force field. *Nucleic Acids Res*, 33(Web Server issue):W382–W388. 15980494[pmid].

[162] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[163] Sun, L., Li, L., Peng, Y., Jia, Z., and Alexov, E. (2017). Predicting protein-dna binding free energy change upon missense mutations using modified mm/pbsa approach: Sampdi webserver. *Bioinformatics*, 34(5):779–786.

[164] Tamura, K., Stecher, G., Peterson, D., Filipski, A., and Kumar, S. (2013). MEGA6: Molecular Evolutionary Genetics Analysis Version 6.0. *Molecular Biology and Evolution*, 30(12):2725–2729.

[165] Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667 – 671.

[166] Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., and Higgins, D. G. (1997). The clustal_x windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 25(24):4876–4882.

[167] Tian, J., Wu, N., Chu, X., and Fan, Y. (2010). Predicting changes in protein thermostability brought about by single- or multi-site mutations. *BMC Bioinformatics*, 11(1):370.

[168] Tovchigrechko, A. and Vakser, I. A. (2006). Gramm-x public web server for protein–protein docking. *Nucleic acids research*, 34(suppl_2):W310–W314.

[169] Trott, O. and Olson, A. J. (2010). Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461.

[170] Utgoff, P. E. (1986). Shift of bias for inductive concept learning. *Machine learning: An artificial intelligence approach*, 2:107–148.

[171] Vaisman, I. I. and Masso, M. (2008). Accurate prediction of stability changes in protein mutants by combining machine learning with structure based computational mutagenesis. *Bioinformatics*, 24(18):2002–2009.

[172] Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13.

[173] Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22.

[174] Vehtari, A., Gelman, A., and Gabry, J. (2017). Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, 27(5):1413–1432.

[175] Veljkovic, V., Cosic, I., Lalovic, D., et al. (1985). Is it possible to analyze dna and protein sequences by the methods of digital signal processing? *IEEE Transactions on Biomedical Engineering*, (5):337–341.

[176] Vishveshwara, S., Brinda, K., and Kannan, N. (2002). Protein structure: insights from graph theory. *Journal of Theoretical and Computational Chemistry*, 1(01):187–211.

[177] Wainreb, G., Ashkenazy, H., Wolf, L., Ben-Tal, N., and Dehouck, Y. (2011). Protein stability: a single recorded mutation aids in predicting the effects of other mutations in the same amino acid site. *Bioinformatics*, 27(23):3286–3292.

[178] Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

[179] Wang, G., Zhao, Y., and Wang, D. (2008). A protein secondary structure prediction framework based on the extreme learning machine. *Neurocomputing*, 72(1):262 – 268. Machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007).

[180] Wegner, P. (1990). Concepts and paradigms of object-oriented programming. *ACM Sigplan Oops Messenger*, 1(1):7–87.

[181] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

[182] Wu, T.-F., Lin, C.-J., and Weng, R. C. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005.

[183] Wu, Z., Kan, S., Lewis, R. D., Wittmann, B. J., and Arnold, F. H. (2019). Machine-learning-assisted directed protein evolution with combinatorial libraries. *arXiv preprint arXiv:1902.07231*.

[184] Yang, H., Parthasarathy, S., and Mehta, S. (2005). A generalized framework for mining spatio-temporal patterns in scientific data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 716–721, New York, NY, USA. ACM.

[185] Yang, K. K., Wu, Z., and Arnold, F. H. (2018). Machine learning in protein engineering. *arXiv preprint arXiv:1811.10775*.

[186] Yoo, I., Alafaireet, P., Marinov, M., Pena-Hernandez, K., Gopidi, R., Chang, J.-F., and Hua, L. (2012). Data mining in healthcare and biomedicine: A survey of the literature. *Journal of Medical Systems*, 36(4):2431–2448.

[187] Zhang, H. (2004). The optimality of naive bayes. *AA*, 1(2):3.

[188] Zhou, H. and Zhou, Y. (2004). Quantifying the effect of burial of amino acid residues on protein stability. *Proteins: Structure, Function, and Bioinformatics*, 54(2):315–322.

[189] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.