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

June 2019

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. ...

Tabla de contenidos

Lista de figuras

Lista de tablas

Chapter 1

Aplicaciones de la minería de datos en ingeniería de proteínas

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. Los estudios se enfocan principalmente, en la evaluación de cambios en la estabilidad de la proteína mediante la variación de energía libre que la mutación provoca [104, 90, 100, 91].

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 [96, 27, 23, 69, 109, 53, 26]. 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 [56, 48], cambios en termo estabilidad [108], propiedades geométricas [9], 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 [26], 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 [89].

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 inspirados 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.

A continuación, se describen los principales conceptos relacionados a minería de datos y 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 Minería de datos

Minería de datos es el proceso de descubrimiento de patrones en set de datos, involucrando métodos asociados a Machine Learning, Estadísticas y sistemas de bases de datos. [59]. La minería de datos es 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. [46, 42]. La minería de datos es el paso de análisis del proceso de *descubrimiento de conocimiento en bases de datos*, o KDD. [45]. Además del análisis en bruto de los datos, también incluye aspectos de manipulación de bases de datos y pre procesamiento de datos, evaluaciones de modelo e inferencia, métricas de interés, consideraciones de complejidad, post procesamiento de estructuras descubiertas, visualización y actualización de la información [14].

En la Figura 2.1, se exponen las principales ramas que componen la minería de datos y los diferentes procesos que se asocian a dichas ramas.

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 [59].

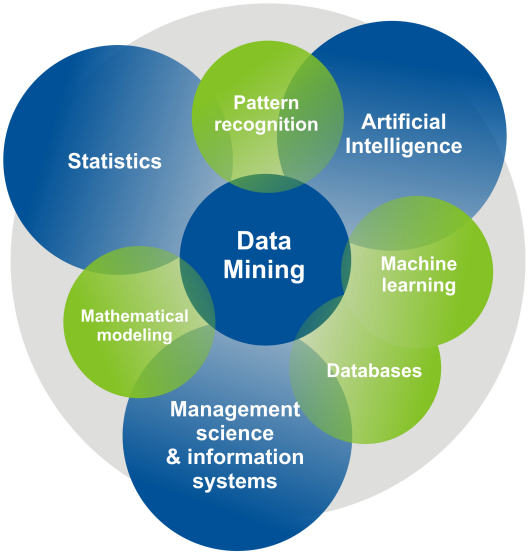


Fig. 2.1 Componentes en la minería de datos

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 [14].

La minería de datos se utiliza en diferentes campos, tales como: genética y genómica [73, 99], ingeniería de proteínas [58, 75, 76], comercio y negocios [63], sistemas de tránsito [79], optimizaciones en procesos industriales [34, 52, 17], reconocimiento de patrones [65, 44], rasgos cuantificables en enfermedades [116, 88, 40] y más recientemente en áreas de dinámicas moleculares [31, 115] y parámetros para la generación de pipe lines automatizados de simulaciones cuánticas en sistemas químicos [82, 36, 98].

2.2 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 [86]. Aplicándose en diferentes campos de investigación: motores de búsqueda [35], diagnósticos médicos [33, 1], detección de fraude en el uso de tarjetas de crédito, bioinformática [72], reconocimiento de patrones en imágenes [43] y textos [87, 4], etc.

Los algoritmos de aprendizaje pueden clasificarse en dos grandes grupos [86]:

- **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.

2.2.1 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 [86].

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.

2.2.2 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.

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	$\min_w \|\|Xw - y\|\|_2^2$	Método más simple, no implica parámetros externos.
Ridge Regression	$\min_w \|\|Xw - y\|\|_2^2 + \alpha\|\|w\|\|_2^2$	Adición de penalización α.
Lasso	$\min_w \frac{1}{2n_{\text{samples}}}\|\|Xw - y\|\|_2^2 + \alpha\|\|w\|\|_1$	Reduce el número de características bajo las cuales depende la solución final.
Elastic-Net	$\min_w \frac{1}{2n_{\text{samples}}}\|\|Xw - y\|\|_2^2 + \alpha\rho\|\|w\|\|_1 + \frac{\alpha(1-\rho)}{2}\|\|w\|\|_2^2$	Usado principalmente en caso de atributos con alta correlación.
Orthogonal Matching Pursuit (OMP)	$\arg\min_\gamma \|\|y - X\gamma\|\|_2^2$ subject to $\|\|\gamma\|\|_0 \leq n_{\text{nonzero\_coefs}}$	Permite fijar el número de coeficientes no nulos.
Bayesian Regression	$p(y\|X,w,\alpha) = \mathcal{N}(y\|Xw,\alpha)$	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.

2.2.3 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 [68]. 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 cartacterística, para así poder comparar esta distancia y evaluar la cercanía para asociarle una clase a este nuevo ejemplo [41]. La distancia a emplear para evaluar la cercanía puede ser: Euclidiana [37], Manhattan [93], coseno [77], Mahalanobis [80], 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 [68].

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*.

KNN utiliza el componente de peso [107], 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 [92].

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

2.2.4 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 [117]. 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 [74, 32, 85]. 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 [85, 67, 81], 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.**

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

2.2.5 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 [49].

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 [15].

A modo de ejemplo, se expone en la Figura 2.2, una representación de un posible árbol, el cual fue desarrollado para entrenar modelos de clasificación utilizando el set de datos iris [47].

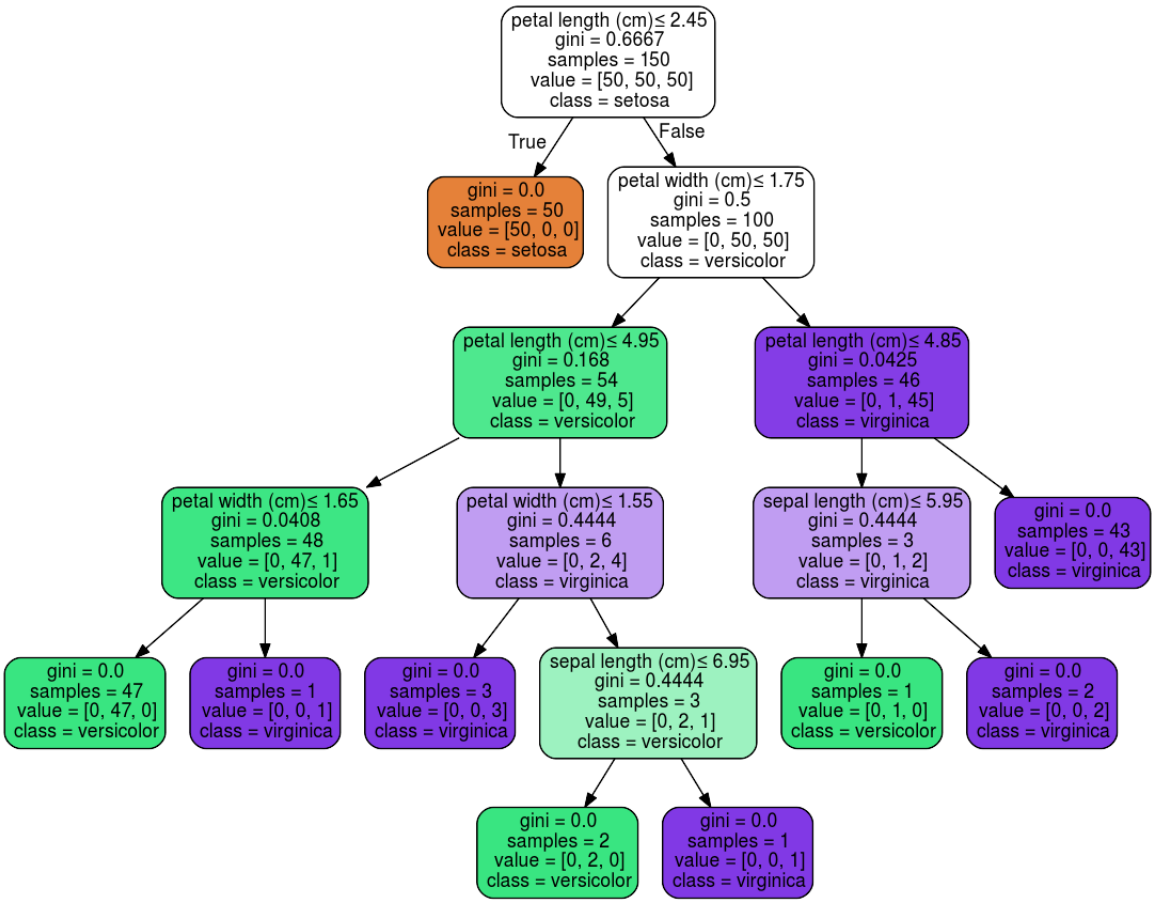


Fig. 2.2 Estructura de árbol para modelo de clasificación de set de datos iris.

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 [22].

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.

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.

2.2.6 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 [103].

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 [103].

Las predicciones se realizan de manera eficiente, utilizando funciones kernel para su transformación en espacios no lineales [5]. 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 [114], estos se calculan utilizando cinco veces una costosa validación cruzada.

Existen diversas variaciones de SVM, tales como: SVC [57], νSVC y LinearSVC, los cuales son capaces de realizar una clasificación multiclase[1] en un conjunto de datos, es 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 [29]. 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.

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

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 2.3:

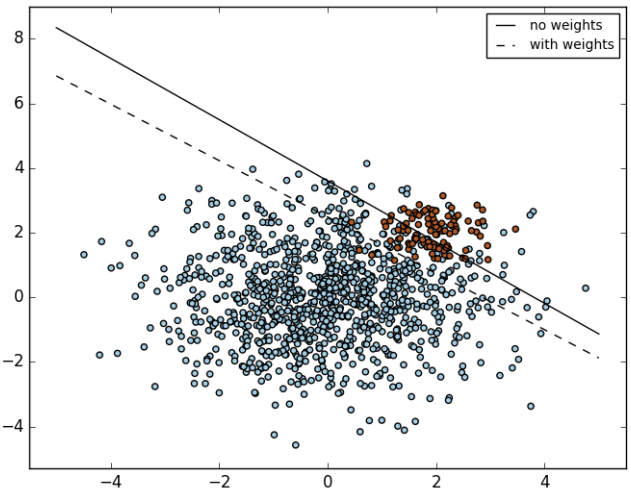


Fig. 2.3 Muestra de desbalance de clases en SVM.

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.

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 2.4

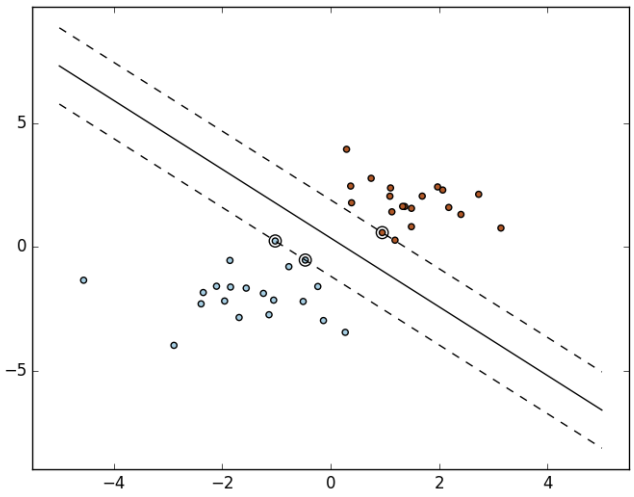


Fig. 2.4 Esquema de hiperplanos en SVM.

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 $x_i \in R^n$ $\varepsilon - SVR$ [105] 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.

2.2.7 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 [39].

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 [71]:

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 [18].

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"* [20].

- Las muestras se extraen con reemplazo, siendo este método conocido como *"Bagging"* [18].

- Los subconjuntos aleatorios del conjunto de datos se basan en subconjuntos aleatorios de las características, esto se conoce como *"subespacios aleatorios"* [8].

- Los subconjuntos aleatorios se crean en base a subconjuntos aleatorios de características y muestras, esto se conoce como *"Random Patches"* [78].

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 [19].

Un esquema representativo del proceso, es como se expone en la Figura 2.5. 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 [21].

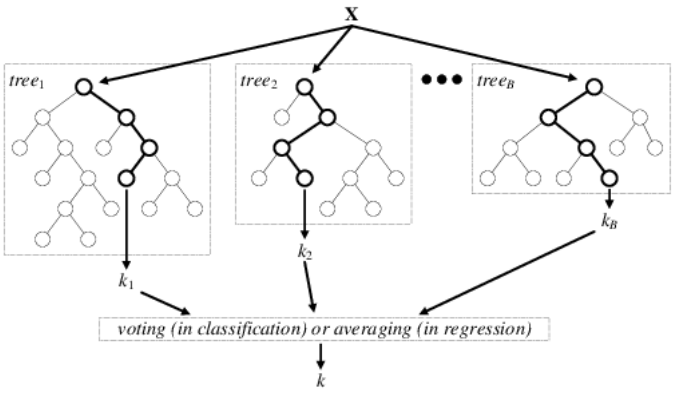


Fig. 2.5 Esquema representativo de algoritmo Random Forest.

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 [25].

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.

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 [60].

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 [50]. 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 [51].

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 [10]. 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.

2.2.8 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 2.6 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.

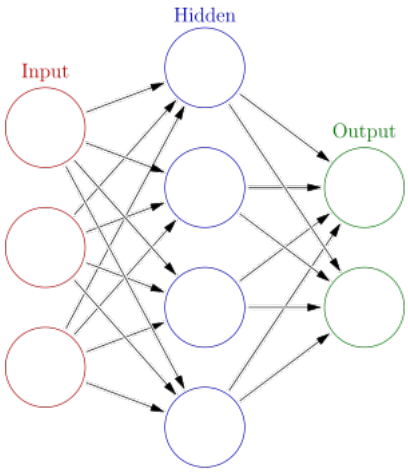


Fig. 2.6 Representación esquemática de una Red Neuronal

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 [12, 11, 38].

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 sistema nervioso, como la codificación neural que intenta definir una relación entre varios estímulos y respuestas neuronales asociadas en el cerebro [11].

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 [38].

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 [7].

Dentro de los principales algoritmos que son utilizados en redes neuronales se encuentran Back Propagation [62] y Multi Layer Perceptron [101].

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.

2.2.9 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 $n-samples$.

2.2.10 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"* [64].

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 [102, 110], siendo las más utilizadas [2]:

- 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 [61], 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 [66]. 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.

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

2.2.11 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 [70].

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 [55].

Dentro de las principales técnicas de validación se encuentra la Validación cruzada con k divisiones [55] 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 [111]. 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 [55].

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 2.7:

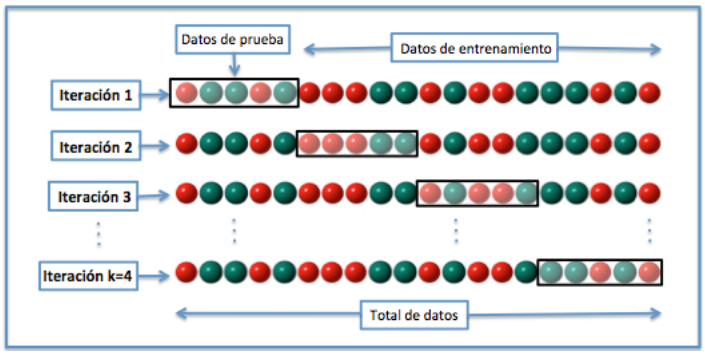


Fig. 2.7 Esquema representativo de validación cruzada.

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 2.8, implicando una mayor cantidad de iteraciones que validación cruzada, provocando además un mayor coste computacional [70].

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.

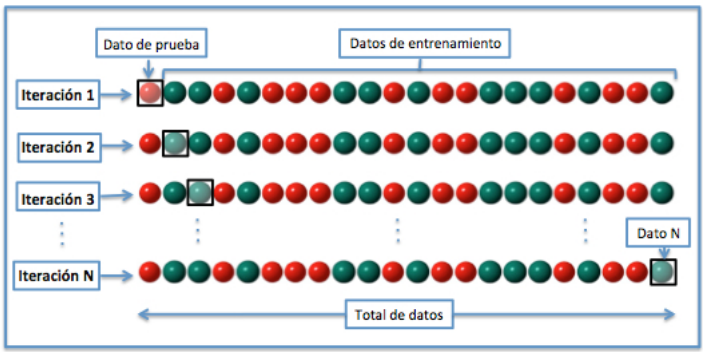


Fig. 2.8 Esquema representativo de Leave One.

2.3 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 [104], Dmutant [118], CUPSAT [91]). 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 [26], utiliza SVM para predecir o clasificar el efecto de la mutación. Mientras que, AUTO-MUTE [84] 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.

2.3.1 Herramientas necesarias para la caracterización de los set de datos

Adicional a las herramientas expuestas, se hace una descripción breve de SDM [90] y MOSST [89], 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

MOSST

2.4 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.

Es posible utilizar técnicas de Meta Learning y algoritmos de aprendizaje supervisado para la generación de modelos de clasificación o regresión de mutaciones puntuales descritas a partir de sus propiedades termodinámicas y filogenéticas?

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?

2.5 Objetivos

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

2.5.1 Objetivo general

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

2.5.2 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 reconocidas.
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.

2.6 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.

2.6.1 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 2.9 se expone un esquema general con los pasos desarrollados para la preparación del set de datos.

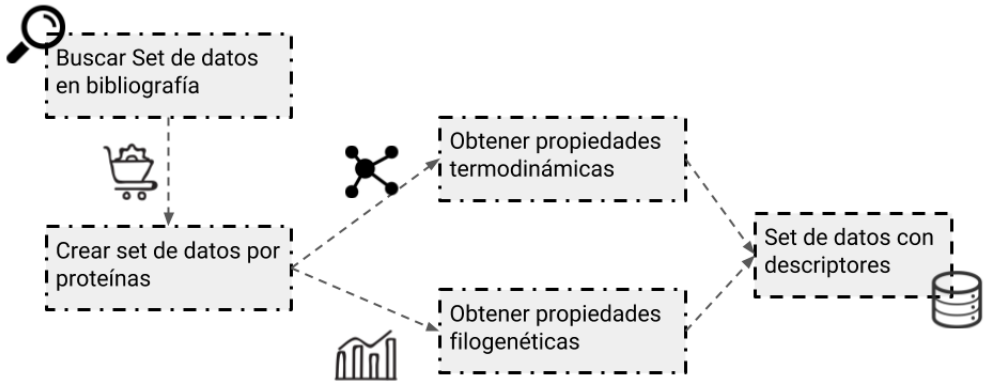


Fig. 2.9 Esquema representativo asociado al proceso de generación de set de datos de mutaciones puntuales en proteínas.

Tal como se expone en la Figura 2.9, 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.

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 2.2.10. En el caso de detectar esta problemática, el conjunto de datos será tratado empleando el método SMOTE (Synthetic Minority Oversampling Technique) [30].

Posteriormente, se aplican las herramientas SDM [90] y MOSST [89] 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 [13] 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 [92]. 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)[3].

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.

2.6.2 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 2.10, 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

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

de los clasificadores o predictores [92], así como Numpy para el uso de módulos estadísticos [113].

Fig. 2.10 Esquema representativo asociado al proceso de creación de meta modelos utilizando la metodología reportada para la herramienta MLSTools (Paper en redacción).

Tal como se observa en la Figura 2.10, 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 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 [92] 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.

2.6.3 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.

2.6.4 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 2.11.

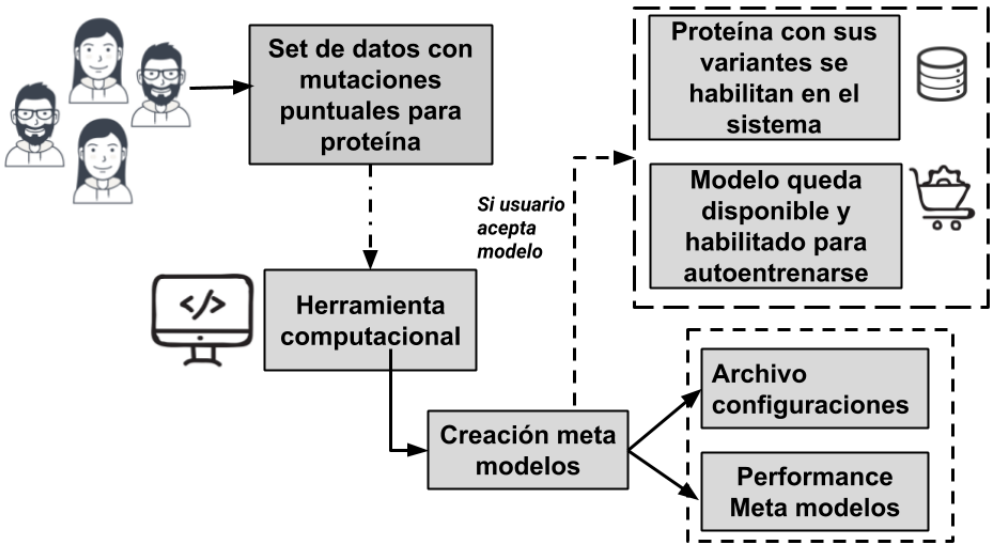


Fig. 2.11 Esquema representativo de flujo asociado a la herramienta de generación de meta modelos para mutaciones puntuales en proteínas de interés.

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.

2.7 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.

2.7.1 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 [112, 106, 94, 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 [6, 24, 28, 97, 26, 95, 69, 83, 54].

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.

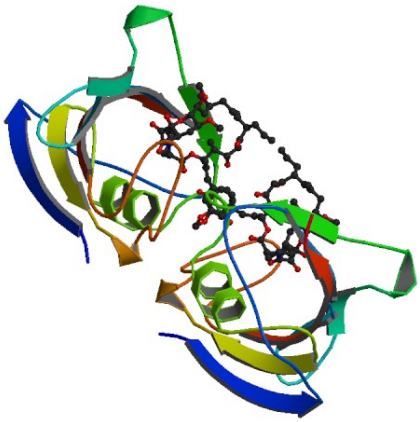
Resumen set de datos de proteínas y sus características				
#	**Código PDB**	**Tipo**	**Ejemplos**	**Descripción**
1.	1A22	Regresión	132	Human growth hormone bound to single receptor
2.	1CH0	Regresión	191	Crystal and molecular structures of the complex of alpha-*Chymotrypsin with its inhibitor Turkey 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

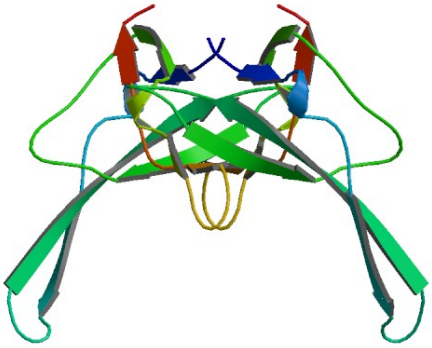
Table 2.9 Resumen de proteínas utilizadas para el desarrollo de meta modelos basados en metodología Meta Learning System propuesta durante este capítulo.

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.

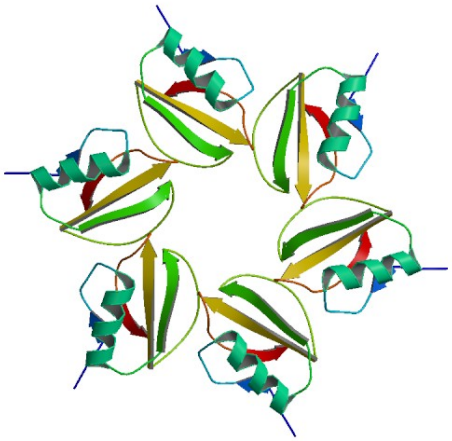
A modo de ilustrar las diferencias estructurales de las proteínas en estudio, en la Figura 2.12 se exponen algunas de las estructuras asociadas a las proteínas utilizadas para desarrollar modelos de clasificación o regresión.



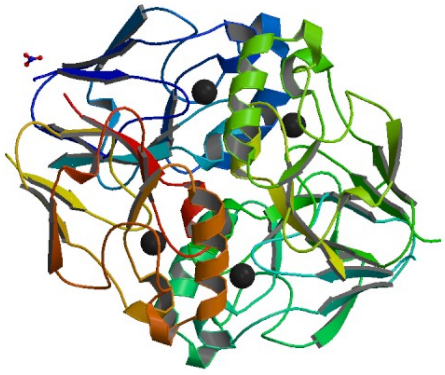
(a) 1FKJ



(b) 1VQB



(c) 2CI2



(d) 5AZU

Fig. 2.12 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

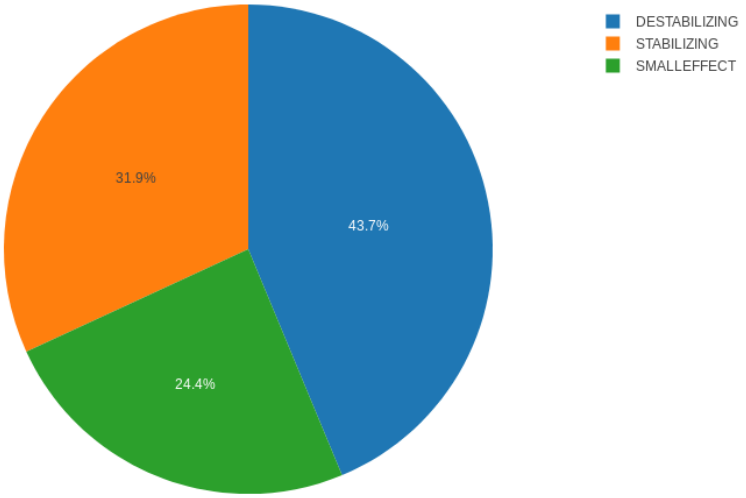
El desbalance de clases se evaluó en aquellos set de datos con respuesta categórica, lo cual contempla, tres posibles casos: Neutral, Estable, No Estable, esto es debido a que todos los estudios donde se evalúan mutaciones, normalmente se analizan cambios que alteren la estabilidad de la proteína. En la Figura 2.13, 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 2.13 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%.

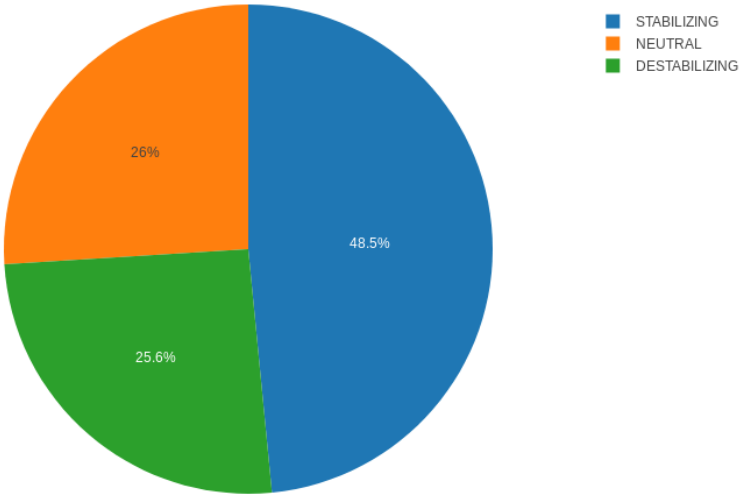
Si bien las proporciones son dispares, para este caso, se considera un desbalance como un elemento que representa menos de un 5% del total de la muestra, además, dado a que la cantidad de ejemplos son elevadas, un 20% o un 25% implica cerca de 200 mutaciones, en promedio, que cumplen dicha característica. 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 [89] 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

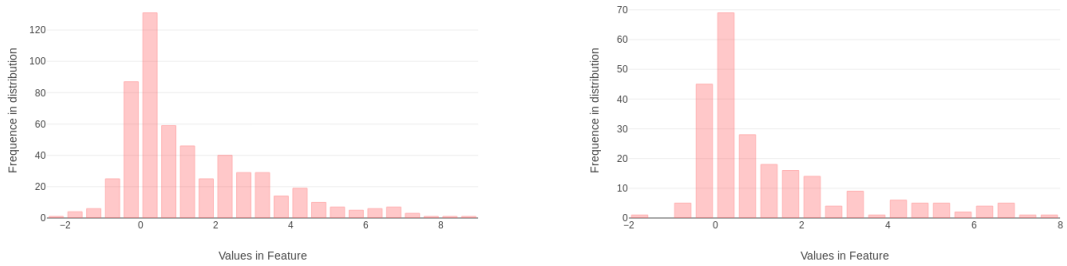


(b) 2RN2

Fig. 2.13 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 2.14.



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

Fig. 2.14 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 [92].

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.

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. 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.

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.

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 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.

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.

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 a

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

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 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.

[1] Ver sección 2.2.10

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 $nx20$ 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.

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 $sx20$ 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. 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.

Se han reportado cerca de 570 propiedades fisicoquímicas que pueden ser utilizadas para describir un residuo en una secuencia lineal de proteínas. 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 espaciales 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

Mutual information

Análisis espaciales de características

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.

Dentro de las principales técnicas de reducción lineales, se encuentran el Análisis de componentes principales (PCA) y sus variantes, tales como: Incremental PCA o Kernel PCA. Además de la aplicación de valores singulares, Factor Analysis, Análisis de Componentes Independientes (ICA) y Factorización de Matrices no Negativas, dentro de las principales.

Cada una de estas técnicas, se describe brevemente a continuación.

Análisis de Componentes Principales (PCA)

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, los cuales se denominan componentes principales, siendo la cantidad menor o igual que las variables originales, donde la principal característica de los componentes principales es que son ordenados en base a la varianza que entregan a los datos, así el primer componente principal aporta una mayor varianza que el segundo y así sucesivamente, basándose principalmente en el uso de vectores propios.

PCA se utiliza principalmente como una herramienta en el análisis de la exploración de datos los cuales tienen como objetivo generar modelos de predicción y sus resultados normalmente se exponen en puntuaciones que tienen estrecha relación con el aporte de varianza que estos entregan

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, esto último es denotado como la reducción de la dimensionalidad en base a los aportes a las varianzas que denotan cada componente.

Definición

Matemáticamente, es posible definir PCA como una transformación lineal ortogonal que transforma 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, $\mathbf{X}$, con una media empírica 0, donde cada una de las filas ($\mathbf{n}$) representan ejemplos y las columnas características o atributos ($\mathbf{p}$).

La transformación está definida por un set de vectores de dimensión $\mathbf{p}$ 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 $\mathbf{t}$ considerado sobre el conjunto de datos sucesivamente heredan la varianza máxima posible de $\mathbf{x}$, con cada carga del vector $\mathbf{w}$.

El primer componente w_1 tiene que 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:

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

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. Sin embargo 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 pxL 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.

Propiedades

Existen tres propiedades claras que facilitan la comprensión y la utilidad del PCA como

medio de manejo de dimensionalidades, y transformación de datos a espacios con coordenadas similares no correlacionados linealmente, éstas son:

- Para cualquier entero $q, 1 \leq q \leq p$, se considera la transformación lineal $y = B'x$, donde y es un elemento del vector q y B es una matriz de (q x p) y la matriz de covarianza para y se puede denotar por $\sum_y = B' \sum B$, entonces la traza de $\sum_y$ es maximizada tomando $B = A_q$ donde A_q es la primera columna q de A.
- Considerando la transformación ortogonal $y = B'x$, la traza de $\sum_y$ es minimizada tomando $B = A_q^*$ donde A_q^* es la última columna q de A.
- Sea la descomposición espectral de $\sum = \lambda_1 \alpha_1 \alpha_1' + ... + \lambda_p \alpha_p \alpha_p'$, se tiene que $Var(x_j) = \sum_{k=1}^{p} \lambda_k \alpha_{kj}^2$.

En resumen, es posible denotar que todas las operaciones de los datos son centradas en cero, tal que $\frac{1}{N} \sum_{i=1}^{N} x_i = 0$, para lo cual se realizan tratamientos sobre la matriz de covarianza, la cual se puede exponer como: $C = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T$, con el fin de obtener los vectores propios, denotados como **eigen descomposition**: $\lambda v = Cv$, los cuales pueden ser reescritos como $\lambda x_i^T = x_i^T Cv \ \forall i \in [1, N]$, lo cual genera la transformación lineal en vectores con menor dimensionalidad expuestos por sus aportes de varianzas en los datos, existentes en espacios con coordenadas similares, sin correlación lineal entre ellos.

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.

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.

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'.

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.

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 Uso de Transformadas de Fourier en digitalización de propiedades fisicoquímicas

3.3 Hipótesis

3.4 Objetivos

3.5 Metodología

Chapter 4

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

Chapter 5

Planificación y estado de avance.

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

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] 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.

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

[15] 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).

[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.

[17] 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.

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

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

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

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

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

[23] 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].

[24] 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.

[25] 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.

[26] Capriotti, E., Fariselli, P., and Casadio, R. (2005). 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].

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

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

[40] 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.

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

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

[43] 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.

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

[45] 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.

[46] 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.

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

[48] 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.

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

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

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

[52] 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.

[53] 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].

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

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

[60] Hastie, T., Rosset, S., Zhu, J., and Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

[61] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[62] 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.

[63] Hofmann, M. and Klinkenberg, R. (2013). *RapidMiner: Data mining use cases and business analytics applications*. CRC Press.

[64] 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.

[65] Jain, A. K., Dubes, R. C., et al. (1988). *Algorithms for clustering data*, volume 6. Prentice hall Englewood Cliffs.

[66] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449.

[67] 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.

[68] 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.

[69] Khan, S. and Vihinen, M. (2010). Performance of protein stability predictors. *Human Mutation*, 31(6):675–684.

[70] 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.

[71] 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.

[72] 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.

[73] Lee, J. K., Williams, P. D., and Cheon, S. (2008). Data mining in genomics. *Clinics in Laboratory Medicine*, 28(1):145–166.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] Maesschalck, R. D., Jouan-Rimbaud, D., and Massart, D. (2000). The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18.

[81] Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] Michie, D., Spiegelhalter, D. J., Taylor, C., et al. (1994). Machine learning. *Neural and Statistical Classification*, 13.

[87] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

[88] Obenshain, M. K. (2004). Application of data mining techniques to healthcare data. *Infection Control &; Hospital Epidemiology*, 25(8):690–695.

[89] 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.

[90] 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].

[91] 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].

[92] 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.

[93] Perlibakas, V. (2004). Distance measures for pca-based face recognition. *Pattern Recognition Letters*, 25(6):711 – 724.

[94] 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.

[95] 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.

[96] 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].

[97] 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.

[98] 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.

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

[104] 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].

[105] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[106] 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.

[107] Tan, S. (2005). Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667 – 671.

[108] 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.

[109] 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.

[110] Van Der Maaten, L., Postma, E., and Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13.

[111] 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.

[112] 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.

[113] 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.

[114] 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.

[115] 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.

[116] 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.

[117] Zhang, H. (2004). The optimality of naive bayes. *AA*, 1(2):3.

[118] 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.