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

Departamento de Ingeniería Química, Biotecnología y Materiales
Universidad de Chile

This dissertation is submitted for the degree of
Doctor of Philosophy

May 2019

Table of contents

List of figures

List of tables

Chapter 1

Aplicaciones de la minería de datos en ingeniería de proteínas

Chapter 2

Desarrollo de modelos predictorios para mutaciones puntuales

El estudio del efecto de mutaciones puntuales en complejos de proteínas, es una de las problemáticas más estudiadas en los últimos años, enfocándose principalmente, en la evaluación de cambios en la estabilidad de la proteína mediante la variación de energía libre que la mutación provoca [62, 53, 58, 54].

Diferentes modelos de clasificación 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 [57, 17, 14, 41, 65, 30, 16]. No obstante, en casos más específicos, se han desarrollado modelos para proteínas exclusivas con el fin de asociar la mutación a un rasgo clínico [32].

En ambos casos, es necesario construir set de datos con respuesta conocida para poder entrenar los modelos 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 [16]. Sin embargo, dejan de lado, los componentes asociados a conceptos filogenéticos y la propensión a cambios de dicha mutación [52].

Dado a los modelos existentes y en vista a la necesidad de generar nuevos sistemas de clasificación para mutaciones puntuales en proteínas, debido al aumento considerable de reportes en los últimos años, se propone una nueva metodología para el diseño e implementación de evaluación de mutaciones puntuales en proteínas, describiéndolas desde los puntos de vista termodinámico y filogenético y usando algoritmos de aprendizaje supervisado como generadores de modelos. Dicha metodología será evaluada en diferentes set de datos de proteínas con mutaciones reportadas con enfoque en estabilidad y estimación de los cambios energéticos. Finalmente, se propone un caso estudio en el cual se aplica la metodología para

generar clasificadores de propensión clínica de mutaciones en la proteína pVHL, asociada a la enfermedad de von Hippel Lindau.

A continuación, se describen algunas herramientas computacionales y su significancia para este estudio a la hora de comparar y analizar los resultados obtenidos, seguido además de los conceptos relacionados al aprendizaje supervisado, junto con la metodología desarrollada.

2.1 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.

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 [50]. Aplicándose en diferentes campos de investigación: motores de búsqueda [20], diagnósticos médicos [19, 1], detección de fraude en el uso de tarjetas de crédito, bioinformática [44], reconocimiento de patrones en imágenes [25] y textos [51, 2], etc.

Los algoritmos de aprendizaje pueden clasificarse en dos grandes grupos [50]:

- **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 [50].

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.

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 K-Vecinos Cercanos

Algoritmo que basa sus estimaciones en la cercanía que presentan los ejemplos entre sí, considerando para una asociación un cierto número de vecinos, denotado como **k**, los cuales presentan características similares a los nuevos ejemplos a estimar. [40]. 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.

Diferentes medidas de distancia son utilizadas para cuantificar la cercanía entre ejemplos y así aplicar la asociación correspondiente, dentro de las métricas más utilizadas se encuentran: Euclidiana [21], Manhattan [56], coseno [45], Mahalanobis [47], entre las principales.

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 [40].

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

[1] Este problema y otros asociados a sistemas de clasificación, son explicados en la sección 2.2.9

KNN utiliza el componente de peso [64], 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 [55].

2.2.3 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 [68]. 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, siendo ejemplos: clasificación de documentos y el filtrado de spam. 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 [49, 39, 48], 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_{yi} 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.**

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

2.2.4 Árboles de Decisión

Se define árbol de decisión como un modelo de predicción, 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. 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 salida en función de diversas variables de entrada [26].

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.

Un árbol de decisión es una representación simple para clasificar ejemplos, el aprendizaje basado en esta metodología es una de las técnicas más eficientes para la clasificación supervisada. Donde cada ejemplo consta de atributos con valores discretos dentro de un dominio de conjunto finito, y existe un sólo término final denominado clasificación. En un árbol de decisión, cada elemento del dominio de la clasificación se llama clase, cada nodo interno (no hoja) está etiquetado con una función de entrada, las ramas procedentes de un nodo etiquetado con una característica están asociados con cada uno de los posibles valores de la característica. Cada hoja del árbol se marca con una clase o una distribución de probabilidad sobre las clases [8].

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 llamada particionamiento recursivo. La recursividad termina cuando el subconjunto en un nodo tienen todos el mismo valor de la variable objetivo, 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 división, esto con el objetivo de seleccionar la mejor partición para un atributo dado, la elección de dicho atributo 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 [13]. Dicha función puede ser expuesta como sigue:

$$\Phi(D,t) = I(t) - \sum_{i=1}^{l} I(t_i)P_i \tag{2.1}$$

Donde:

- $I(t)$ representa la Medida de Impureza asociada al nodo **t**, desde el cual se comenzará a realizar la partición o nodo padre.
- $\sum_{i=1}^{l} I(t_i)$ representa la suma ponderada de las impurezas de los nodos hijos t_i generados a partir de una división **D**.
- P_i representa la proporción de ejemplos que siguen la rama **i** asociada a la división **D**.

Dentro de las medidas de impuerza, existen:

- Gini Index = $\sum\ p_i\ x\ (1 - p_i)$
- Entropía = $-\sum\ p_i\ x\ \log_2(p_i)$

Siendo la más utilizada la medida de Entropía [27].

En ambas, p_i corresponde a la proporción de ejemplos asociados a cada una de las clases, presentes en el nodo evaluado.

El algoritmo para el cual se entrena y se clasifica un nuevo ejemplo es el que sigue:

- Partir desde un nodo inicial o padre.
- Seleccionar el mejor atributo que divide de una manera óptima los ejemplos, lo cual se observa por medio de la función de ganancia de información.
- Se clasifica los ejemplos del conjunto de entrenamiento de un nodo entre sus descendientes.
- El proceso finaliza si los ejemplos del conjunto de entrenamiento quedan perfectamente clasificados, esto ocurre en dos casos: todos los ejemplos pertenecen a una misma clase o se llega a una hoja.
- En el caso de no cumplirse lo del punto anterior, se itera para cada rama de manera recursiva, utilizando sólo los ejemplos que llegan a esa rama.

2.2.5 Máquina Soporte de Vectores

Máquina soporte de vectores (SVM por sus siglas en inglés), son modelos de aprendizaje supervisados asociados al análisis de los datos utilizados para la clasificación. Se construye un modelo que asigna nuevos ejemplos a una categoría u otra, conviertiéndolo en un clasificador binario lineal no probabilístico [61].

Un modelo SVM es 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 [61].

SVM puede realizar eficientemente una clasificación no lineal utilizando funciones kernel [3], con el fin de generar transformaciones de espacio dimensional de los datos, para mapear implícitamente sus entradas en espacios característicos de alta dimensión.

Las ventajas de máquinas de soporte vectorial son:

- Efectivo en espacios de dimensiones altas.
- Efectivo aún en casos donde el número de dimensiones es mayor que el número de muestras.
- 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 del núcleo pueden ser especificadas para la función de decisión. Se proporcionan núcleos comunes, pero también es posible especificar núcleos personalizados.

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.
- SVMs no proporciona directamente estimaciones de probabilidad [67], estos se calculan utilizando cinco veces una costosa validación cruzada.

Existen diversas variaciones de SVM, tales como: SVC [33], NuSVC y LinearSVC, los cuales son capaces de realizar una clasificación multiclase[2] en un conjunto de datos, es decir, ya no depender de un clasificador único para dos clases.

SVC presenta una aplicación basada en libsvm [18]. 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. El apoyo multiclase es manejado según un esquema de uno vs uno.

Por otro lado NuSVC presenta características similares a SVC, pero, utiliza un parámetro para controlar el número de vectores de soporte. La aplicación se basa en libsvm.

LinearSVC es similar a SVC pero, se utiliza una función de kernel lineal, además es implementado en términos de liblinear en lugar de libsvm, por lo que tiene más flexibilidad en la elección de las penalizaciones y las funciones de pérdida y debería escalar mejor a un gran número de muestras. Esta clase soporta entradas densas y escasas y el soporte de multiclase se maneja de acuerdo con un esquema de uno contra el resto.

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 etapa de entrenamiento, la principal característica es que se utilizan vectores de apoyo para el set de entrenamiento, los que son denominados

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

vectores de soporte, normalmente se utilizan funciones kernel para la obtención de estos vectores de soporte.

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 problema 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 2.1:

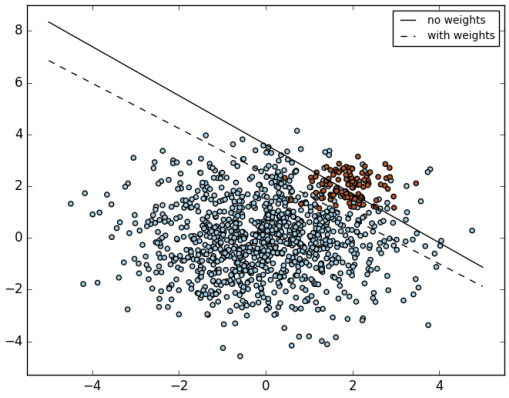

Fig. 2.1 Muestra de desbalance de clases en SVM.

Complejidad

Las máquinas de soporte vectorial son herramientas poderosas, pero sus requerimientos de computación y almacenamiento aumentan rápidamente con el número de vectores de entrenamiento. El kernel de un SVM es un problema de programación cuadrática (QP), separando los vectores de soporte del resto de los datos de entrenamiento. Es posible eslacar esta solución entre $O(n_{features} \times n_{samples}^2)$ y $O(n_{features} \times n_{samples}^3)$.

Formulación Matemática

SVM construye un hiperplano o conjunto de hiperplanos en un espacio dimensional alto o infinito, que puede usarse para clasificación, regresión u otras tareas. Intuitivamente, se logra una buena separación por el hiperplano que tiene la mayor distancia a los puntos de

datos de entrenamiento más próximos de cualquier clase (llamado margen funcional), ya que en general, cuanto mayor es el margen, menor es el error de generalización del clasificador, tal como se expone en la Figura 2.2:

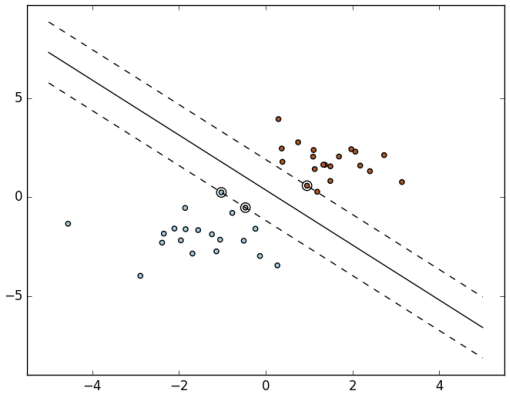

Fig. 2.2 Esquema de hiperplanos en SVM.

Se expone la formulación matemática para cada uno de los clasificadores expuestos anteriormente:

SVC

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:

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

NuSVC

Se introduce el parámetro ν el cual controla el número de vectores de soporte y errores de entrenamiento. El parámetro $\nu \in (0,1]$ es un límite superior en la fracción de errores de entrenamiento y un límite inferior de la fracción de vectores de soporte.

SVR

Dados los vectores de entrenamiento $x_i \in R^n$ $\varepsilon - SVR$ [63] 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:

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

2.2.6 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 [23].

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 [43]:

1. Métodos ponderado, 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, Random Forest.

2. Métodos boosting: basados en la construcción secuencial de modelos, intentando disminuir el sesgo del modelo combinando, cumple con la filosofía *"la unión de varios modelos débiles, puede construir uno fuerte"*. Ejemplos comunes de esto son AdaBoost, 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 [9].

Existen diferentes formas de formar 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"* [11].
- Las muestras se extraen con reemplazo, siendo este método conocido como *"Bagging"* [9].
- Los subconjuntos aleatorios del conjunto de datos se basan en subconjuntos aleatorios de las características, esto se conoce como *"subespacios aleatorios"* [5].
- Los subconjuntos aleatorios se crean en base a subconjuntos aleatorios de características y muestras, esto se conoce como *"Random Patches"* [46].

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 [10].

Un esquema representativo del proceso, es como se expone en la Figura 2.3. 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 [12].

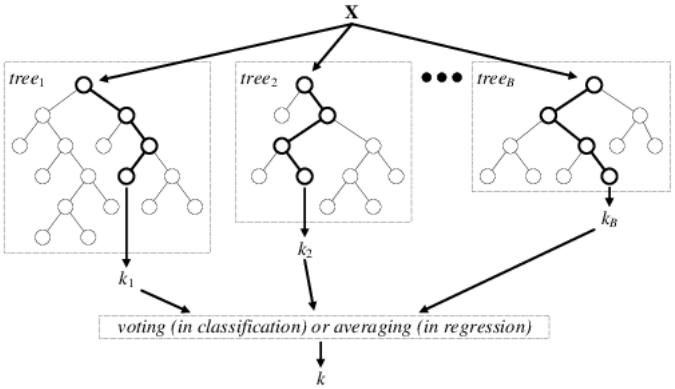

Fig. 2.3 Esquema representativo de algoritmo Random Forest.

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 [15]. 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 [34]. 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 [28]. 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 [29].

2.2.7 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 imagen 2.4 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.

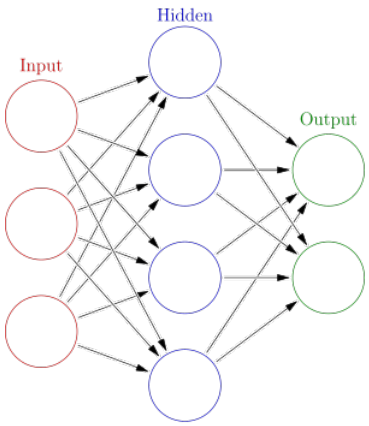

Fig. 2.4 Representación esquemática de una Red Neuronal

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 [7, 6, 22].

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 [6].

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 [22].

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 [4].

Dentro de los principales algoritmos que son utilizados en redes neuronales se encuentran Back Propagation [36] y Multi Layer Perceptron [59].

2.2.8 Medidas de desempeño

Medir el desempeño del modelo predictivo es importante a la hora de evaluar qué tan efectivo es el entrenamiento y 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, 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, cuántos 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 para 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 para 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 tp corresponde a verdaderos positivos y fp a los falsos positivos y 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 $n-samples$.

2.2.9 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"* [37], es decir si existen *n* ejemplos y la cantidad de atributos es *n x n* es posible que ocurra dicha problemática, para solucionar este problema, existen técnicas asociadas a reducción de dimensionalidad [60, 66], siendo las más utilizadas métodos lineales de reducción como PCA y derivados, Análisis de características

basados en modelos de clasificación/regresión aplicando Random Forest, técnicas probabilísticas asociadas al Mutual Information y evaluación de características relacionadas mediante coeficientes de Pearson o matrices de Correlación[3]. En forma similar, también existe 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 ejemplo, técnicas como Mutual Information, Análisis de Correlaciones son bastante utilizadas en este problema. Otro posible problema que se puede denotar es el sobreajuste [35], 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 o regresión se basa en el desbalance de clases o en la tendencia hacia rangos específicos de valores en métodos de regresión [38]. 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 falso positivo. 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.

2.2.10 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. Sin embargo, 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, esto permite aumentar la generalización del modelo, debido a que las medidas de desempeño varían de levemente de aplicación en aplicación y al considerar dicha técnicas, se tienen distribuciones de las medidas del modelo, reportando siempre, la media de dicha distribución. Dentro de las principales técnicas de validación se encuentra la Validación cruzada con k divisiones y un caso particular conocido como *Leave one out*, en el

[3]Estas técnicas, se explican en el capítulo ***Digitalizando propiedades fisicoquímicas de proteínas a partir de su secuencia lineal***

cual el valor de k es igual a la cantidad de ejemplos en el entrenamiento. 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 [31]. En un problema de predicción, a un modelo se le suele asignar un conjunto de datos, de los datos conocidos sobre los que se ejecuta el entrenamiento (conjunto de datos de formación) y un conjunto de datos desconocidos (o primeros datos) 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 como sobre ajuste.

La idea es dividir el set de datos totales abarcando un set de entrenamiento y un set de validación, lo cual se puede explicar en la Figura 2.5:

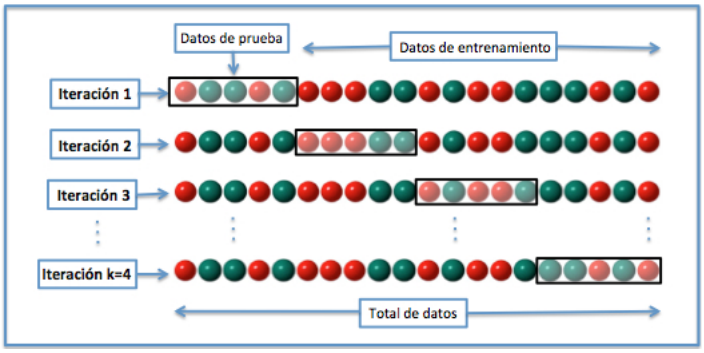

Fig. 2.5 Esquema representativo de validación cruzada.

Una ronda de validación cruzada implica dividir una muestra de datos en subconjuntos complementarios, realizar el análisis en un subconjunto (denominado conjunto de entrenamiento) y validar el análisis en el otro subconjunto (denominado conjunto de validación o conjunto de pruebas). Para reducir la variabilidad, varias rondas de validación cruzada se realizan utilizando diferentes particiones, y los resultados de validación se promedian durante las rondas, siendo las más utilizadas *10-fold cross validation*.

Leave one out (Dejar uno)

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 2.6, implicando una mayor cantidad de iteraciones que validación cruzada, provocando además un mayor coste computacional [42].

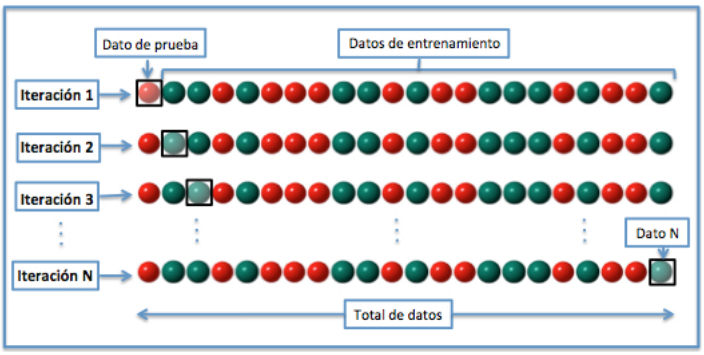

Fig. 2.6 Esquema representativo de Leave One.

2.2.11 Meta Learning

2.3 Hipótesis

En base a las herramientas existentes y en vista del aumento considerable de datos asociados a mutaciones en proteínas y el conocimiento de las respuestas que éstas generan, se evidencia la necesidad del desarrollo de herramientas computacionales o nuevos modelos de clasificación o regresión que faciliten el entrenamiento de proteínas singulares y la evaluación de sus mutaciones puntuales, con el fin de poder evaluar nuevos ejemplos y cuáles serían los efectos de estos, sin tener que entrar 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 clasificación de nuevos ejemplos?

2.4 Objetivos

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

2.4.1 Objetivo general

Diseñar e implementar estrategias de 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.

2.4.2 Objetivos específicos

Dentro de los objetivos específicos se encuentran los siguientes.

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

2.5 Metodología propuesta

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

2.5.1 Preparación de set de datos

La preparación del set de datos consiste en obtener data para poder entrenar los modelos de clasificación, la data se asocia a información de mutaciones en proteínas y la respuesta que ésta genera. En la Figura 2.7 se expone un esquema general con los pasos desarrollados para la preparación del set de datos.

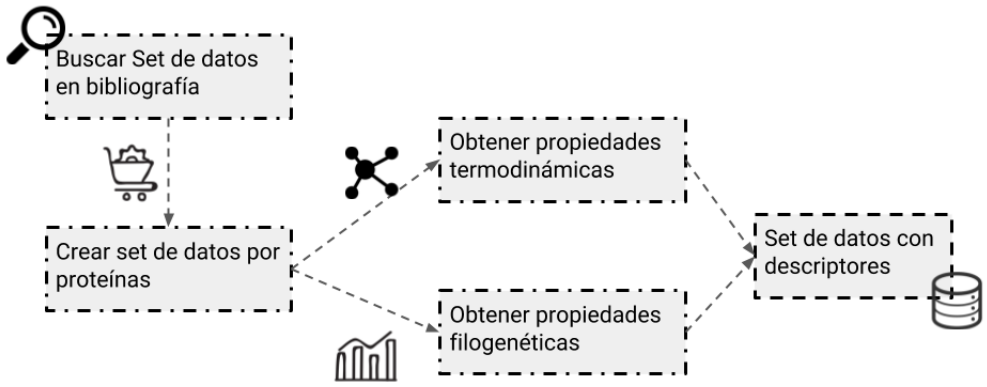

Fig. 2.7 Esquema representativo asociado al proceso de generación de set de datos de mutaciones puntuales en proteínas.

Tal como se expone en la Figura 2.7, 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, scripts fueron implementados 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, eliminando aquellos ejemplos cuyo valor no superara el 5% del total de ejemplos.

Posteriormente se aplican las herramientas SDM [53] y MOSST [52] con el fin de obtener los descriptores asociados a las propiedades termodinámicas y filogenéticas. Para ello, scripts 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.

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 [55]. 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).

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 mantiene las respuesta con el fin de demostrar la robustez del método y la eficacia de éste sin importar el tipo de modelo que se éste entrenando.

2.5.2 Implementación de meta modelos de clasificación/regresión

La implementación de meta modelos consiste en la obtención de un grupo de estimadores que en conjunto, permiten clasificar o predecir nuevos ejemplos. Para ello, se diseña e implementa una metodología basada en Meta Learning System y aplicando conceptos similares a los utilizados por los métodos de ensamble en la etapa de evaluación del desempeño del clasificador.

En la Figura 2.8, 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 **Acá debo citar el paper del MLSTraining Tool :D**. Cada una de las etapas contempla un conjunto de scripts implementados en lenguaje de programación Python y

empleando la librería Scikit-Learn para el entrenamiento y evaluación de los clasificadores o predictores [55].

Fig. 2.8 Esquema representativo asociado al proceso de creación de meta modelos utilizando Meta Learning System Tools.

Tal como se observa en la Figura 2.8, es posible identificar etapas claves en el proceso: Exploración de modelos, Selección y Generación de los meta clasificadores/predictores. 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 2.1 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 MLSTraining					
#	**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
8.	Multi layer perceptron	Redes neuronales	Función de activación Solver Taza de aprendizaje Alpha Shuffle Máximo iteraciones	Clasificación y Regresión	2160
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	**2886**

Table 2.1

Como se observa en la Tabla 2.1, son sobre 2800 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 permite entregar 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, considerando un valor $k = 10$, con el fin de poder disminuir posibles sobreajustes.

Selección de modelos

Cada distribución de medida de desempeño perteneciente los modelos entrenados en la fase de Exploración, se somete a test estadísticos 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 cortes 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 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 $k = 10$ para validar los modelos.

Con el fin de evaluar el desempeño de los meta clasificadores, 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, es decir, los modelos son del tipo regresión, 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.

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

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

Nuevos ejemplos pueden ser clasificados o predecir su respuesta, dependiendo sea el caso, a partir de los meta modelos desarrollados. En el caso de estimadores basados en variables continuas, los nuevos ejemplos se someten a cada uno de los modelos individuales pertenecientes al sistema, los cuales generan una respuesta individual, a partir de dichas respuestas, se

genera un intervalo de confianza con un nivel de significancia $\alpha = 0.05$ donde existe una mayor probabilidad de que se encuentre el valor real de la predicción. Para ejemplos que impliquen clasificación, se obtiene la respuesta de cada modelo individual y se evalúa la probabilidad de ocurrencia de cada categoría, entregando así, la respuesta condicionada por una probabilidad de ocurrencia del evento.

2.5.4 Uso de meta modelos en sistemas de proteínas

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

Un flujo general del uso de la herramienta, se expone en la Figura 2.9.

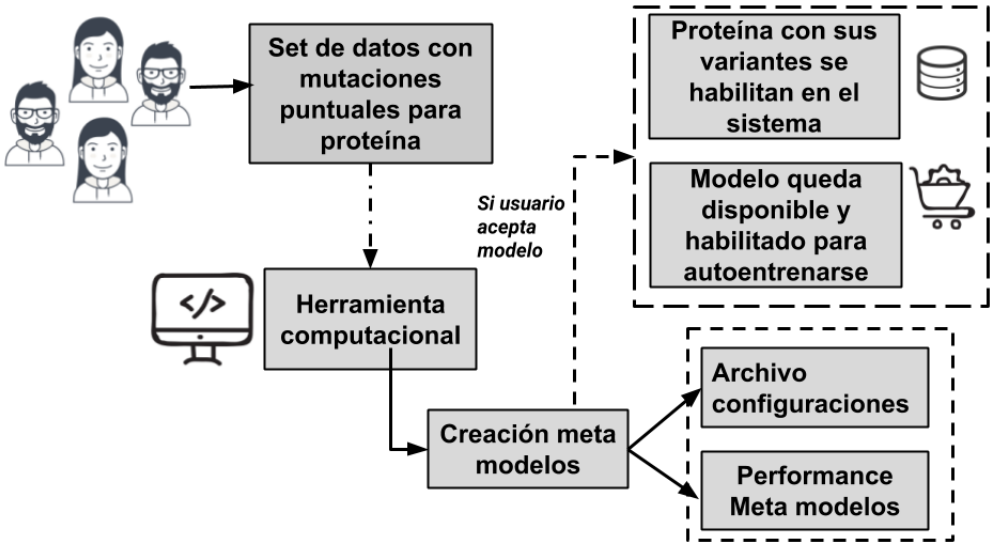

Fig. 2.9 Esquema representativo de flujo asociado a la herramienta de generación de meta modelos para mutaciones puntuales en proteínas de interés.

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 System. 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. 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 por medio de un consentimiento informado, 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.

Chapter 3

Codificación de secuencias lineales de proteínas

Chapter 4

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

Chapter 5

Modelamiento matemático discreto aplicado al estudio de estructuras de proteínas.

Chapter 6

Reconocimiento de patrones y extracción de información en sistemas complejos multi-dimensionales

Chapter 7

Un caso de estudio completo: Aplicación de técnicas de minería de datos y reconocimiento de patrones para modelar el sistema de interacción antígeno anticuerpo

References

[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] 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.

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

[4] 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.

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

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

[7] 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.

[8] 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).

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

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

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

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

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

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

[15] 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.

[16] 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].

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

[18] 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.

[19] 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.

[20] 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.

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

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

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

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

[25] 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.

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

[27] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

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

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

[30] Getov, I., Petukh, M., and Alexov, E. (2016). 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].

[31] 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.

[32] 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.

[33] 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.

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

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

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

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

[49] 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.

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

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

[52] 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.

[53] 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].

[54] 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].

[55] 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.

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

[57] Quan, L., Lv, Q., and Zhang, Y. (2016). Strum: structure-based prediction of protein stability changes upon single-point mutation. *Bioinformatics*, 32(19):2936–2946. 27318206[pmid].

[58] 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.

[59] 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.

[60] 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.

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

[62] 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].

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

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

[65] 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.

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

[67] 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.

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