Alberto Blanco Álvarez

MODELO DE MACHINE LEARNING PARA PREDECIR EL GRADO DE INHIBICIÓN DE UN COMPUESTO CONTRA LA PROTEASA PRINCIPAL DEL COVID-19

TRABAJO DE FIN DE GRADO

Dirigido por el Sr. Jordi Duch Gavaldà

Grado de Biotecnología e Ingeniería Informática

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Agradecimientos

Me gustaría agradecer Gerard Pujadas y a Santiago García por acogerme dentro del grupo de investigación Covid-Moonshot, durante el periodo de tiempo entre febrero y junio de 2021, para formarme en el ámbito de la investigación biotecnológica aplicada a la informática. Me integró en el grupo como uno más y me siento en deuda, pues me ha permitido aprender la metodología, habilidad, destreza y conocimientos que se requieren para enfrentarse a esta línea de investigación.

Quisiera expresar también mi gratitud a mi tutor, Jordi Duch Gavaldà, quien me ha guiado en todo momento. Por su esfuerzo, dedicación y ayuda, que sin duda han enriquecido mi trabajo, y que me ha aconsejado en todo momento para poder completar de forma exitosa esta difícil tarea.

También, quisiera agradecer a mis tutores dentro del grupo de investigación, Alberto Fernández y Francesc Serratosa, los cuales, junto a Santiago y Gerard, me han encaminado y ayudado en cualquier momento de necesidad para poder completar toda tarea asignada dentro del periodo de prácticas, y por motivarme para aprender todo lo posible de las herramientas existentes que permiten unir el mundo de la biotecnología con el mundo de la informática.

Finalmente, quisiera agradecer, de todo corazón, a mis padres y a mi hermano, los cuales han confiado en mí en todo momento de mi estancia universitaria y me han apoyado de forma incondicional en todas las etapas de mi vida. Desde lo más profundo de mi corazón, muchas gracias. Por todo.

Resumen

Este proyecto se ha desarrollado con la intención de poder agilizar las investigaciones actuales respecto al tema del COVID-19 y el desarrollo de una vacuna eficaz contra el virus. La idea principal consiste en la creación de un modelo de *Machine learning* que reciba como parámetro de entrada descriptores específicos de un sistema inhibidor de la proteasa principal del COVID-19 y calcule, a partir de ellos, la concentración necesaria de esa sustancia para alcanzar un grado de inhibición de la proteasa equivalente a un 50%, también conocido como pIC50. El objetivo final consiste en poder desplegar, en un futuro, este modelo en una plataforma online que sea accesible mediante la web. Los resultados obtenidos fueron aceptables tras probar diversos tipos de modelos, decantándose finalmente por un modelo de regresión lineal.

Resum

Aquest projecte s'ha desenvolupat amb la intenció de poder agilitzar les investigacions actuals pel que fa a el tema de l'COVID-19 i el desenvolupament d'una vacuna eficaç contra el virus. La idea principal consisteix en la creació d'un model de Machine learning que rebi com a paràmetre d'entrada descriptors específics d'un sistema inhibidor de la proteasa principal de l'COVID-19 i calculi, a partir d'ells, la concentració necessària d'aquesta substància per arribar a un grau d'inhibició de la proteasa equivalent a un 50%, també conegut com pIC50. L'objectiu final consisteix a poder desplegar, en un futur, aquest model en una plataforma en línia que sigui accessible mitjançant la web. Els resultats obtinguts van ser acceptables després de provar diversos tipus de models, decantant-se finalment per un model de regressió lineal.

Abstract

This project has been developed with the intention of speeding up the current research in the subject of COVID-19 and the development of an effective vaccine against the virus. The main idea consists of creating a Machine learning model that receives as the input parameter specific descriptors of a COVID-19 main protease inhibitor system and calculates, from them, the necessary concentration of that substance to achieve a degree of protease inhibition equivalent to 50%, also known as pIC50. The goal is to be able to deploy, in the future, this model on an online platform that is accessible through the web. The results obtained were acceptable after testing various types of models, finally opting for a linear regression model.

Índice

Índice de tablas

Índice de figuras

1 Introducción

1.1 Descripción

En la industria farmacéutica y en el ámbito de la investigación se hacen uso de simulaciones en computadores de alto rendimiento para poder encontrar o desarrollar moléculas que se puedan emplear para el desarrollo de nuevos fármacos, analizar las estructuras tridimensionales de sistemas proteicos y modificarlos para obtener fármacos de segunda generación y la validación de estos *a posteriori*. Sin embargo, obtener altos niveles de conocimiento a nivel molecular de la intervención en procesos bioquímicos y biofísicos de una proteína consiste en una ardua tarea, la cual produce elevados costos temporales y económicos en lo que constituye una "competición" entre las diferentes empresas en crear un método para desarrollar fármacos más efectivos y con la mayor velocidad posible.

Además, juntando las condiciones actuales de investigación con la pandemia, y las consecuencias que ha originado en el mundo laboral el virus, todos los esfuerzos dentro del ámbito farmacéutico se han juntado con el fin de elaborar vacunas contra el COVID-19.

Con el objetivo de poder agilizar todo el proceso de obtención de fármacos contra el virus, conocido como el proceso de *Drug discovery*, el grupo de investigación de la URV conocido como Covid-Moonshot plantea, como solución a este problema, el diseño de un software basado en inteligencia artificial, empleando el lenguaje Python en la totalidad del sistema.

En el proceso de diseñar un modelo adecuado que sea capaz de predecir el pIC50 (grado de inhibición de la proteasa equivalente a un 50%) esperado de un sistema, introduciendo los descriptores intramoleculares de un sistema que actúe de inhibidor para la proteasa principal del COVID-19 (M-Pro), la cual, inhibida, restringiría la acción viral del coronavirus. Estos descriptores son introducidos vía dos ficheros .csv, obtenidos por dos herramientas diferentes, *glide_dock_XP*, software de Schrödinger, y CADD_ASFP, decidiendo emplear los descriptores obtenidos por ambos al tratar con una pequeña muestra de proteínas para realizar el modelo (107 sistemas para entrenar y testear el modelo).

Otra parte importante de este trabajo consistió en la prueba de múltiples algoritmos y modelos de *machine learning* junto a diferentes técnicas de reducción de dimensiones en los datos (como PLS) para encontrar un modelo y un conjunto de características que permitan obtener un modelo adecuado para predecir el grado de inhibición de diversos compuestos.

1.2 Objetivos

Los objetivos académicos de este trabajo consisten en adquirir conocimiento sobre el mundo de *machine learning*, repasar diversos modelos y algoritmos existentes y mejorar los modelos obtenidos adquiridos mediante diversas metodologías de refinamiento de datos.

Partiendo que se hace uso de la metodología de *Supervised Machine Learning*, se pretende obtener datos nuevos a partir de los anteriores y no clasificar los datos en conjuntos, se emplearán modelos predictivos y algoritmos de regresión. Para esto se dividirán los datos iniciales en dos conjuntos, en un conjunto de entrenamiento, que será el que prepare el modelo especificado, empleando un algoritmo, para que pueda predecir de forma correcta un nuevo valor, y un conjunto de prueba, que será empleado para validar el modelo generado y comprobar que la funcionalidad sea correcta.

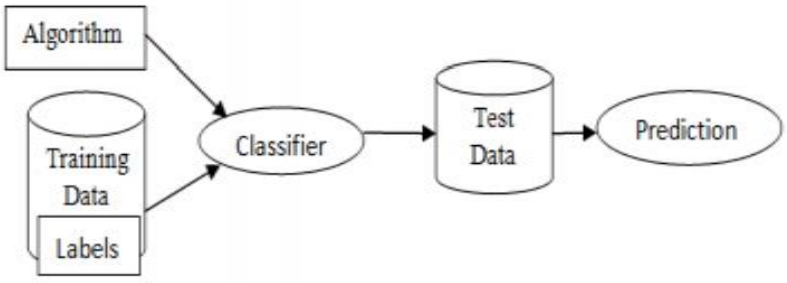

Figura 1: Machine Learning Supervisado

Siguiendo esta línea de trabajo, los objetivos se pueden dividir en 3 segmentos diferenciados:

1) Análisis de diversos algoritmos de *machine learning*: Para poder elaborar un modelo adecuado con capacidad suficiente para poder calcular el pIC50 de un sistema a analizar frente a la proteasa principal del COVID-19, se ha de emplear un algoritmo adecuado, que se ajuste adecuadamente a los datos introducidos, y proponga predicciones para el grado de inhibición que cause cada sistema en el COVID-19 relativamente precisas. Para esto, se ha de estudiar el comportamiento, funcionamiento y finalidad de diversos tipos de algoritmos, aprovechando las ventajas de cada modelo, y comparar los resultados obtenidos por cada modelo.
2) Comparación de la funcionalidad de cada algoritmo: Una vez se hayan seleccionado los diferentes algoritmos a evaluar, se prepararán sistemas con los cuales se pueda evaluar cada modelo, comprobando cuan efectivo resulta el modelo estudiado para nuestro conjunto de datos. Para comparar los resultados, se usarán métricas de puntuación, que califican la eficacia del modelo, como el coeficiente de correlación de la regresión, o el MSE. Se emplearán, adicionalmente, técnicas que evalúen la eficacia de cada modelo de forma individual (técnicas de *cross-validation*) y se compararán los valores obtenidos adicionalmente.
3) Mejora del modelo: Tras comparar los modelos estudiados y optar por uno de todos los algoritmos para proseguir con el programa, se procederá a emplear diversas técnicas de *tuning*, ajustando los parámetros empleados en el conjunto

de entrenamiento, buscando aumentar el grado de correlación entre los datos predichos y los datos reales. Para ello, se emplearán algoritmos de ajuste de modelos de regresión, como la regresión *stepwise*, para determinar qué características son determinantes en el momento del cálculo del pIC50 a predecir.

1.3 Background

1.3.1 Background teórico sobre modelos

Para todo estudio en el que se emplean modelos de *machine learning* se posee una gran cantidad de datos, que han de ser analizados y modelados. Para poder realizar un estudio exitoso, el algoritmo empleado en el proceso tiene una elevada importancia en el momento de determinar el producto final, por lo que seleccionar un algoritmo adecuado a emplear en los *pipelines* de *machine learning* que se ajuste a los datos introducidos. Los modelos de regresión predicen un resultado, conocido como la variable dependiente (el pIC50, en el caso de este estudio) a partir de una relación entre las variables independientes.

Con este objetivo, se han estudiado diversos algoritmos y modelos de *machine learning* orientados a la predicción de valores mediante regresión:

1.3.1.1 Regresión lineal

Un modelo de regresión lineal está caracterizado por mostrar un ratio creciente o decreciente constante, encontrando tendencias lineales en los datos. En otras palabras, el valor de la variable dependiente cambia siguiendo un patrón constante.

La relación entre las variables empleadas en el cálculo de la variable dependiente se refiere como regresión lineal en la función lineal de la función primaria, siendo este modelo el más usado estadísticamente.

Este tipo de regresiones son descritas siguiendo la siguiente fórmula:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon} \qquad (1)$$

En la fórmula 1, y es la variable independiente, que puede ser tanto continua, como categórica, x es una variable dependiente, que tendrá un valor continuo. En el caso de que se emplee más de una variable independiente, se presentará en la ecuación más términos similares a $X\beta$, los cuales describan la relación que exista entre cada otra variable con el modelo.[22,23,24]

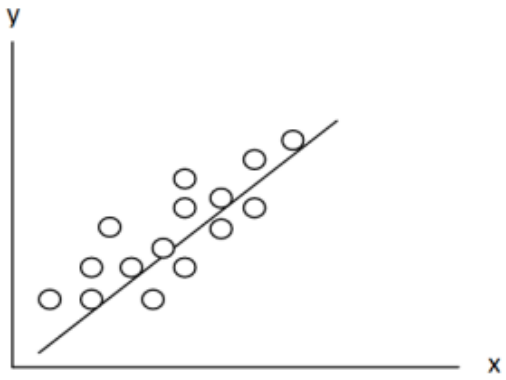

Figura 2: Regresión lineal simple.

1.3.1.2 Regresión polinómica

Un modelo de regresión polinómica es un modelo que, al igual que la regresión lineal, busca una relación entre las variables independientes y la variable independiente para poder calcular una fórmula en la cual los coeficientes tengan un grado máximo de 2 o superior.

Este suele ser adecuado de emplear tras probar que un modelo de regresión lineal no se ajusta a los datos (al ocurrir, de manera frecuente, *underfitting*).

De regla general, se genera una ecuación con el grado que se le indique al algoritmo, añadiendo nuevos términos polinómicos a la regresión lineal para obtener una regresión polinómica.

Las regresiones polinómicas siguen la siguiente ecuación:

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + b_3 x_1^3 + \ldots . b_n x_1^n \qquad (2)$$

En la fórmula 2 se describe la fórmula que emplean los algoritmos de regresión polinómica, donde y es la variable dependiente, b es una variable independiente, y x el término polinómico por el que se multiplica, determinando el grado de la regresión. En el caso de emplear varias variables independientes, se multiplicará la variable en cuestión por el coeficiente correspondiente al grado.

Se emplean de forma general en conjuntos de datos que estén ordenados de forma no lineal.

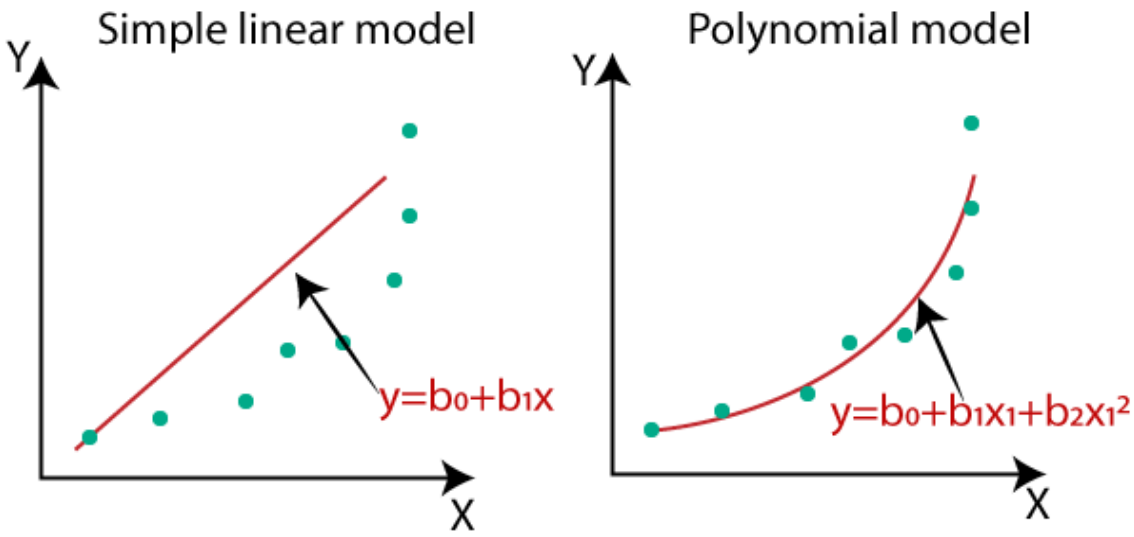

Figura 3: Comparación entre modelo lineal y polinómico.

En la Figura 3 se puede visualizar cómo un modelo polinomial puede ajustarse mejor a los datos que un modelo lineal, resultandos útiles para encontrar modelos que siguen ecuaciones matemáticas complejas.[20,21]

1.3.1.3 Regresión de árbol de decisión.

Un árbol de decisión es una estructura de datos en forma de árbol que consiste en un número arbitrario de nodos, en las cuales se sitúa diversas ramas en cada nodo. Un nodo del cual 'crecen' otros nodos se denomina nodo interno. El resto de nodos se denominan hojas. Para la regresión, se emplea una función de *fitting*, la cual divide los datos en dos o más grupos en un nodo interno.

El estimulante de un árbol de decisión es el algoritmo que genera el árbol a partir de una instancia. Este algoritmo busca crear el árbol a partir de la división del *dataset* en cada nodo, calculando un valor de *fitness* para cada variable (de acuerdo con la función de *fitting* seleccionada). En cada nodo, el algoritmo calcula el error entre el valor predicho y el valor real de la variable dependiente a calcular, y tras comparar los valores, la variable que posea el valor de *fitness* menor se selecciona como punto de división, generando así nuevas ramas, ejecutándose de forma recursiva el algoritmo, dividiendo el conjunto de datos en sets más pequeños mientras se desarrolla el árbol, hasta finalizar de dividir el conjunto de datos y el crecimiento del árbol.

Los árboles de decisión son técnicas que permiten tanto clasificar como predecir valores, permitiendo realizar tanto modelos de clasificación como modelos de regresión. Además, ofrecen otras ventajas:

- Proveen flexibilidad para tratar un amplio rango de tipos de respuesta, desde categorías hasta números.
- Son simples de entender y evaluar.
- Permiten validar un modelo realizando tests estadísticos, a la vez que permiten calcular valores en el caso de falta de algún valor. Además, permiten la entrada de varias características de forma simultánea.

Sin embargo, también presenta una desventaja crucial, que pueden resultar inestables, ya que pequeñas variaciones en los datos resulten en nuevos nodos, y llegue a causar *overfitting*.

Se han de revisar los parámetros de los árboles de decisión, ya que estos pueden influenciar la calidad de la estimación. Estos parámetros engloban la función de *fitness*, la profundidad del árbol, la división de la muestra, el muestreado de las hojas, y el número de características (las características que se añadan de forma inicial, a mayor cantidad de características, mayor ruido).

La función de fitness busca minimizar el error entre el valor predicho y el valor observado. La profundidad determina cuanto puede crecer el árbol, a mayor profundidad, más información se tendrá sobre los datos. La división de la muestra determina el mínimo número de muestras para formar un nuevo nodo. El muestreado de las hojas determina el mínimo número de muestras necesarias para crecer una hoja.[19]

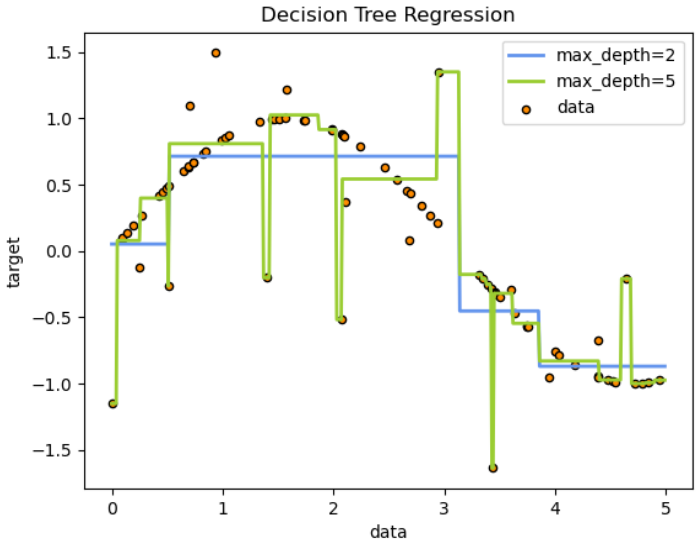

Figura 4: Regresión de árbol de decisión.

1.3.1.4 Regresión de procesos Gaussianos

Un proceso Gaussiano es un proceso estocástico, una colección de variables aleatorias indexadas por tiempo o espacio, en las cuales cada colección finita de esas variables aleatorias tiene una distribución normal multivariable, una combinación finita linear en la que están normalmente distribuidas. La distribución de un proceso Gaussiano consiste en la distribución conjunta de todas las variables aleatorias, siendo como tal, una distribución sobre funciones en un dominio continuo.

Los modelos de *machine learning* que emplean procesos Gaussianos hacen uso de la técnica conocida como *lazy learning*, siendo este un método de aprendizaje en el cual, la generalización del conjunto de entrenamiento se retrasa hasta que se realiza una petición al sistema.

Los modelos de regresión de procesos Gaussianos miden la similitud entre diferentes puntos, usando funciones kernel, para predecir el valor de una variable dependiente. Esta predicción no consiste únicamente en una estimación, sino que también posee información sobre su incerteza, tratándose de una distribución Gaussiana

de una dimensión. Estos se tratan de modelo no lineales, que vienen especificados por la siguiente ecuación, que depende de su función media (m(x)) y su función de covariancia o kernel (k(x,x')).

$$f(x) \sim GP\left(m(x), k(x, x')\right) \quad (3)$$

En un modelo de regresión tradicional, se infieren los datos empleando una única función. Sin embargo, en las regresiones de procesos Gaussianos, emplea un proceso Gaussiano frente a una función. Inicialmente se define únicamente el kernel, dando una distribución de antemano a la función. Este kernel describe la forma general de las funciones.

$$f(x) \sim \mathbf{N}(\mathbf{0}, \mathbf{K}) \quad (4)$$

Una vez se haya obtenido una distribución inicial, se emplea un conjunto de datos de entrenamiento, y se enfrenta la función con los datos de entrenamiento frente a un proceso Gaussiano, obteniendo una distribución conjunta.

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_* \\ K_*^T & K_{*,*} \end{bmatrix}\right) \quad (5)$$

La función del modelo consiste en obtener una función f* para un conjunto de variables independientes, permitiendo realizar de esta forma predicciones, calculando el condicional posterior a los datos observados, y obtener una distribución Gaussiana que describa el grado de incerteza del dato predicho. [15,16,17,18]

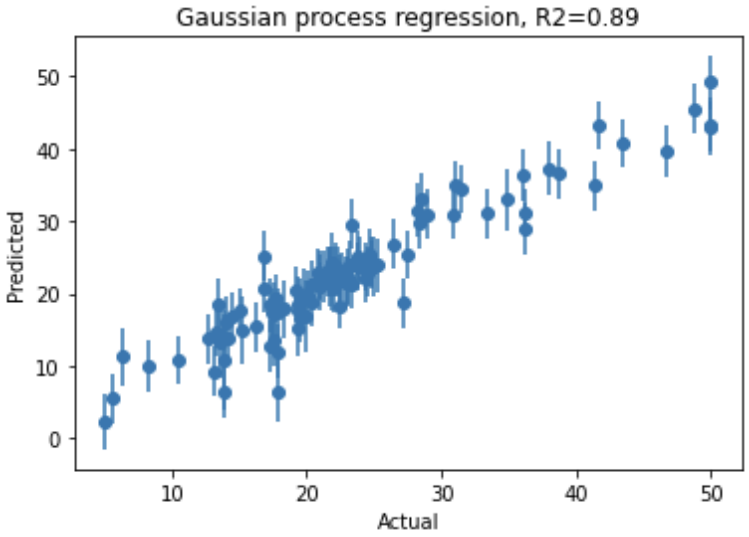

Figura 5: Regresión de procesos Gaussianos.

1.3.1.5 Regresión de cresta

La Regularización de Tikhonov, también conocida como regresión de cresta (o *Ridge regression*), consiste en un método de *machine learning* empleado principalmente para analizar conjuntos de datos que presentan multicolinearidad.

La multicolinealidad es la relación de dependencia lineal fuerte (entendiéndose como una dependencia fuerte un elevado coeficiente de correlación) entre más de dos variables explicativas en una regresión múltiple que incumplen el supuesto de Gauss-Markov cuando es exacta. En otras palabras, es la correlación alta entre más de dos variables explicativas.

$$\boldsymbol{A \cdot x = b} \qquad (6)$$

$$\boldsymbol{||A \cdot x - b||^2 + ||r \cdot x||^2} \qquad \textbf{(7)}$$

Esta regresión busca, para una ecuación tal como (6), donde no hay solución única para x, la regresión de cresta minimiza la ecuación (7) para encontrar una solución, en la que r es la matriz de Tikhonov definida por el usuario que permite al algoritmo preferir soluciones más determinadas sobre otras que se puedan elaborar.

La regresión de cresta busca proveer una respuesta a problemas que no tienen una solución determinada, introduciendo información adicional para seleccionar la mejor solución posible para el problema.

Este método sirve para minimizar la suma de los residuales al cuadrado, el primer término de (7), donde || representa la norma euclidiana, es decir, la distancia del origen al vector resultante.

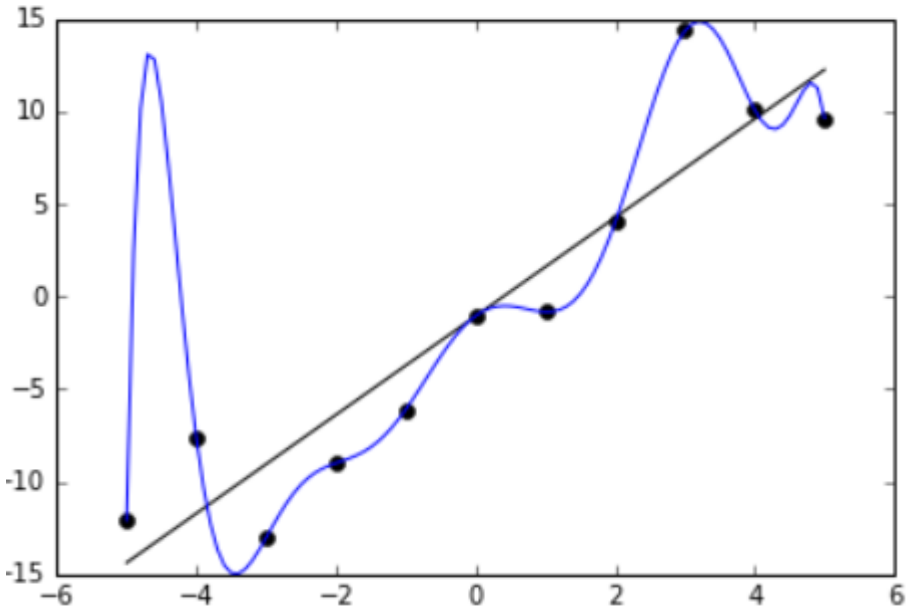

Figura 6: Regresión de crestas.

El algoritmo minimiza el error que aparezca entre los diferentes puntos correspondientes a los datos. Sin embargo, no generaliza de forma adecuada, causando, de forma general, un sobreajuste de los datos (*overfitting*).

Este tipo de regresiones previene *overfitting* y *underfitting* mediante la introducción de un término de penalización, $||r \cdot x||^2$. Empleando mayores valores de r, resulta en valores menores para x, disminuyendo los efectos de *overfitting*. De todas formas,

valores elevados pueden causar *underfitting*, que previene al método de encajar adecuadamente los datos. De forma típica, se emplea un múltiplo de la matriz identidad como valor de r, efectivo para evitar el *overfitting*.

Una de las ventajas principales que destaca este tipo de modelos es que resultan particularmente eficientes en la resolución de problemas en las que se tienen aproximadamente 100 muestras o menos o cuando se posee un mayor número de parámetros que de muestras, por lo que puede resultar productivo en el problema que se quiere tratar.[13,14]

1.3.1.6 Regresión PLS

La regresión de mínimos de cuadrados parciales o PLS *regression* es un conjunto de métodos estadísticos que reducen los predictores a un conjunto más pequeño de componentes no correlacionados y realiza una regresión de mínimos cuadrados sobre estos componentes, en lugar de hacerlo sobre los datos originales. Entre los métodos asociados, cabe destacar el algoritmo de modelado Path-PLS, NIPALS-PCA y NIPALS-CCA.

La idea básica del PLS es la de reducción de la dimensión en regresión múltiple, con la garantía de que los primeros componentes ortogonales mejoran la predicción.

Este algoritmo reduce el número de predictores mediante el empleo de una técnica similar a las del análisis de componentes principales para extraer un conjunto de componentes que describa la correlación máxima entre los predictores (las variables independientes empleadas) y las variables de respuesta. Se puede calcular tantos componentes como el número de predictores que haya, aunque con frecuencia se utiliza la validación cruzada para identificar el conjunto más pequeño de componentes que provea la mayor capacidad predictiva.

Para determinar los componentes principales, se emplea una combinación lineal de las variables originales para calcular un nuevo conjunto de variables no correlacionadas entre sí, disminuyendo las variancias entre ellas de forma progresiva.

$$y_j = a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jp}x_p = a'_{jx} \qquad (8)$$

En la fórmula (8) $x_1, x_2 \ldots$ representan las variables originales, a_j' representa un vector de constantes.

El primer componente se calcula eligiendo a_1 de modo que y_1 tenga mayor varianza posible, sujeta a la restricción $a_1' a_1 = 1$. El segundo componente se calcula obteniendo a_2 de modo que la variable obtenida, y_2 careza correlación con y_1, y asi consecutivamente con el resto de componentes deseados.

Una de las principales ventajas que presenta PLS es que resulta especialmente útil cuando los predictores empleadas para la predicción son muy colineales entre sí o cuando se poseen una mayor cantidad de predictores que de observaciones, resolviendo una de las principales fallas de la regresión de mínimos cuadrados ordinarios, ya que produce coeficientes con elevados errores estándar.

PLS no presupone que los predictores son fijos, por lo que pueden medirse con error, resultando en un aumento de la robustez del modelo frente a la incertidumbre de las mediciones.[9,10,11]

1.3.1.7 Red Neuronal

Una red neuronal es una red formada por neuronas (nodos), componentes básicos que actúan como modelos matemáticos interconectados, con el objetivo de alcanzar estructuras con arquitecturas similares al sistema nervioso humano, constituyendo una de las estructuras básicas de los modelos de *Deep learning*, *machine learning* en la que se emplean varias capas de aprendizaje para el modelo.

Están compuestas por capas de nodos, que se dividen en una capa de *input*, una o varias capas ocultas, y una capa de *output*. Cada nodo o neurona artificial se conecta a otra, teniendo asociada un peso y umbral determinados. Si la salida de un nodo individual supera el umbral especificado, el nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no envía datos a la siguiente capa.

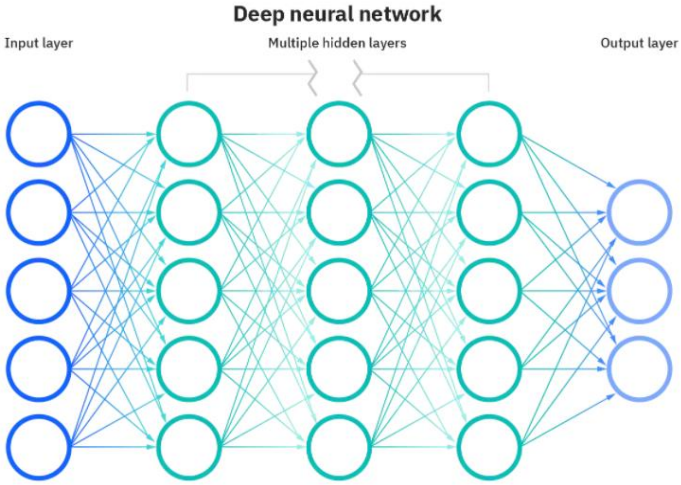

Figura 7: Red neuronal.

Cada nodo individual tiene su propio modelo de regresión lineal. Están compuestos por los datos de entrada, los pesos, el umbral (*bias*) y el output, obteniendo la siguiente fórmula:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

$$\text{output} = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

Figura 8: Fórmula de los nodos de una red neuronal.

Donde w es el dato de entrada, y x es el peso.

Una vez una capa de entrada se determina, se asignan los pesos, que sirven para determinar la importancia de una variable y su contribución. Tras realizar los cálculos, si el output supera el umbral, se activa el nodo, pasando datos a la siguiente capa que actuarán como input, lo que constituye el proceso de *feedforward*.

Para evaluar la eficiencia de una red neuronal, se emplea una función de coste, o una función de pérdida, usándose de forma común el MSE (*Mean Squared Error*).

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$

Figura 9: Cálculo del MSE.

El objetivo es minimizar la función de coste para ajustar los pesos y los umbrales empleados, hasta encontrar mínimos locales en una capa, que serán empleados como input para las siguientes capas.[12]

1.3.1.8 TPOT

TPOT (*Tree-Based Pipeline Optimization Tool*) es una herramienta desarrollada por AutoML orientada a *data science* con el objetivo de automatizar la construcción de *pipelines* de *machine learning* mediante la combinación de un árbol de expresión flexible, que representa los pipelines con algoritmos de búsqueda estocásticos, como la programación genérica.

TPOT representa una herramienta para el aprendizaje automático de modelos de *machine learning* muy potente, de código abierto, capaz de descubrir eficientemente modelos de elevado rendimiento para un determinado conjunto de datos.

En la implementación se emplean tres tipos de operadores:

- *Supervised Classification Operators*: Guardan las predicciones de las clasificaciones como nuevas características y la clasificación del pipeline.
- *Feature Preprocessing Operators:* Modifican el *dataset* y retornan el resultado de la modificación.
- *Feature Selection Operators:* Reducen el número de características en el *dataset* empleando algún criterio y retornan el *dataset* modificado.

Estos operadores se combinan en un *machine learning pipeline*, empleándose para construir árboles de programación genérica, usado para realizar las predicciones. Cada *pipeline* ha de tener, como resultado final, un clasificador como operando final, para poder construir el árbol, proveyendo, así, una representación flexible de los *pipelines* de *machine learning*.

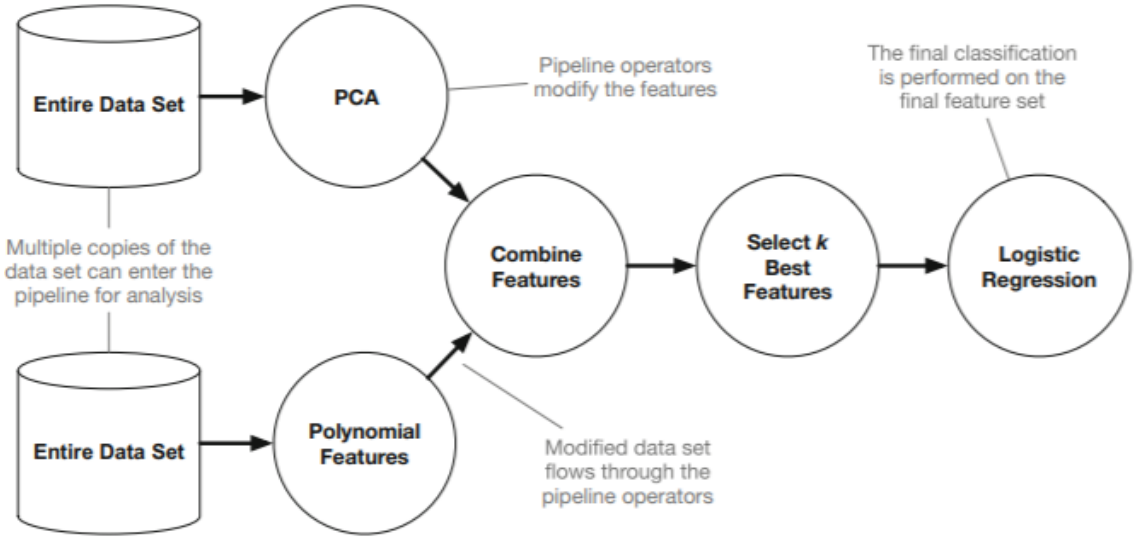

Figura 10: Tree-based pipeline, TPOT.

En esta figura, cada círculo corresponde un operador de *machine learning* y las flechas indican la dirección del flujo de datos.[5.6,7,8]

1.3.1.9 Regresión Stepwise

La regresión *Stepwise* consiste en la construcción iterativa paso a paso de un modelo de regresión en el que se realiza una selección de variables independientes a ser usadas en un modelo final.

El modus operandi de la *Stepwise regression* consiste en la adición y eliminación en sucesión de potenciales variables explicativas, y en la prueba de la significancia estadística tras cada iteración.

El objetivo de este tipo de regresiones consiste en, mediante el empleo de una serie de tests (como F-tests o t-tests), encontrar un conjunto independiente de variables que influencian, de forma significativa, al cálculo de la variable dependiente en un modelo. Esto se realiza, de forma computacional, mediante iteraciones. Se puede lograr probando variables independientes de forma individual, e incluirlas en el modelo de regresión si son estadísticamente significantes, o mediante la inclusión de todas las variables potencialmente independientes y la consiguiente eliminación de todas que no sean estadísticamente independientes.

Existen tres tipos de aproximaciones:

- *Forward selection*: Comienza sin variables en el modelo, y comprueba, de forma individual, cada variable, añadiéndola al modelo en el caso que sea estadísticamente significante, repitiendo el proceso hasta que los resultados sean óptimos.
- *Backward elimination*: Comienza con un conjunto de variables independientes, eliminando de forma individual una y comprobando si la variable eliminada es estadísticamente significante.
- *Bidirectional elimination*: El resultado de la combinación de los dos métodos anteriores.

El empleo de este algoritmo resulta eficaz en sistemas en los que se poseen un número elevado de variables independientes, con números reducidos de muestras, eliminando variables que puedan resultar en una disminución de la efectividad de un modelo, facilitando la tarea de encontrar patrones para los modelos.[4]

1.3.2 Descripción de las herramientas empleadas

1.3.2.1 Entornos de trabajo

1.3.2.1.1 **Anaconda**

Anaconda es una distribución libre y abierta de los lenguajes Python y R habitualmente usado en la ciencia de datos y en *machine learning* para el procesamiento de grandes volúmenes de información, realizar análisis predictivos y cómputos científicos, para simplificar el despliegue y administración de los paquetes software. Esta distribución incluye paquete orientados a la ciencia de datos, apropiado para Windows, Linux y macOS.[29]

Se ha empleado esta distribución ya que proporciona una multitud de herramientas adecuadas para la realización de trabajos relacionados con el *machine learning* y proyecto de *Data science*.

1.3.2.1.2 Jupyter Notebook

Jupyter Notebook se trata de un entorno de programación interactivo en web diseñado para crear documentos de blocs de notas de Jupyer. Estos son documentos JSON que siguen un esquema de versionado, conteniendo una lista ordenada de celdas de input y output las cuales puede contener código (de Python en este caso), texto, esquemas, plots gráficos y diversos tipos de media.

Se han empleado esta herramienta para poder separar en el momento de programar cada segmento de código en diversas celdas que se puedan probar de forma aislada, permitiendo así ordenar el código y pudiendo realizar pruebas aisladas.

1.3.2.2 Tecnologías, lenguajes y algoritmos

1.3.2.2.1 **Python**

Python es un lenguaje de programación interpretado de alto nivel, y de propósito general. Se caracteriza por tener una legibilidad muy fácil. Es paradigma porque permite programar de forma secuencial, orientada a objetos o de forma funcional.

En este proyecto se emplea como único lenguaje de programación, en conjunto a diversas librerías, para obtener los datos de los sistemas inhibitorios, preparar el modelo de *machine learning* y predecir la concentración necesaria para alcanzar un 50% de inhibición para nuevos compuestos.

1.3.2.2.2 **Scikit learn**

También conocido como Sklearn, es una librería de software de *machine learning* para Python que posee diversos algoritmos de clasificación, clustering y regresión, constituyendo una herramienta clave para el desarrollo de modelos en Python.[27,28]

Se ha empleado esta librería en el desarrollo de los modelos ya que ofrece una gran variedad de algoritmos y utilidades para programar un modelo y estructurar un sistema de análisis de datos.

1.3.2.2.3 Pandas

Pandas es una librería de software para Python empleado para la manipulación de datos y análisis. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series de tiempo.

Se emplea Pandas en este proyecto para conseguir las características de los sistemas y el pIC50 de cada uno de estos a partir de un fichero .csv, que será el input del usuario, obteniendo así el conjunto de datos total que será dividido en test y train.

1.3.2.2.4 Scipy stats

Consiste en un módulo perteneciente a la librería de Python de Scipy, la cual es una biblioteca de Python con herramientas y algoritmos de uso matemático. Este módulo contiene un elevado número de conjuntos de distribuciones probabilísticas y funciones estadísticas.[25]

Se emplea para obtener el coeficiente p de cada una de las características frente al valor del pIC50 final, para reducir el número de características empleadas a aquellas con las que se tenga correlación al valor final.

1.3.2.2.5 Statsmodels

Statsmodels es un paquete de Python que permite a los usuarios explorar datos, estimar modelos estadísticos y realizar tests estadísticos, complementando al módulo de SciPy de estadísticas de Scipy.

Se emplea para poder comprobar los valores que asumen cada característica en el momento de predecir el valor del pIC50 en un sistema, junto a otros parámetros de un modelo, como el coeficiente de correlación, entre otros.

1.3.2.2.6 Matplotlib y pyplot

Matplotlib es una librería de plotting para Python que permite realizar diversos tipos de grafos en extensión a la librería NumPy. Provee una API orientada a objetos para realizar plots en aplicaciones. Pyplot es un módulo en el paquete de matplotlib que proporciona una interfaz para generar figuras y ejes de forma implícita y automática.[26]

Se han empleado para representar los datos obtenidos de las predicciones frente a los datos reales, y poder comparar la eficacia de los modelos de forma gráfica.

1.3.2.2.7 OLS

También conocido como regresión de mínimos cuadrados ordinarios, OLS es una regresión de tipo lineal que permite identificar la relación entre un input o conjunto de atributos y una variable que dependa de los anteriores, siendo este método capaz de estimar los coeficientes que asumirán los diversos atributos en la regresión lineal.

Se ha empleado OLS para analizar el modelo lineal final por el que se opta, y conocer el coeficiente de correlación y los coeficientes de los atributos en la regresión lineal.

1.3.2.2.8 Leave-One-Out Cross-Validation

El método de *Leave-One-Out Cross-Validation* (LOOCV) es un método iterativo que se inicia empleando como conjunto de entrenamiento todas las observaciones disponibles excepto una, que se excluye para emplearla como validación. Si se emplea una única observación para calcular el error, este varía mucho dependiendo de qué observación se haya seleccionado. Para evitarlo, el proceso se repite tantas veces como observaciones disponibles, excluyendo en cada iteración una observación distinta, ajustando el modelo con el resto y calculando el error con dicha observación. Finalmente, el error estimado por el LOOCV es el promedio de todos los errores calculados.

Este permite reducir la variabilidad que se origina si se divide aleatoriamente las observaciones únicamente en dos grupos. Esto es así porque al final del proceso de LOOCV se acaban empleando todos los datos disponibles tanto como entrenamiento como validación. Al no haber una separación aleatoria de los datos, los resultados de LOOCV son totalmente reproducibles.

En este estudio se emplea para evaluar cada uno de los modelos finales a comparar y poder tomar una decisión sobre el modelo que mejor se ajuste a nuestro sistema de datos.

2 Estudio de los modelos y selección final

Lo primero que se realizó en el estudio consistió en la obtención de los diversos sistemas que se emplearán para crear un modelo capaz de cumplir el objetivo propuesto.

Para esto, se investigó en diversos papers de la comunidad científica métodos, algoritmos y herramientas que pudieran servir para describir, mediante variables numéricas, la unión entre la proteína que se vaya a emplear y la proteasa principal del COVID-19 que se quiere inhibir. El primer objetivo consiste en obtener diversos descriptores capaces de indicar cómo se realiza esta unión de fragmentos proteicos, el receptor y el ligando, para poder realizar métodos computacionales con ellos.

Se probaron diversas herramientas, pero las dos empleadas en el estudio final consistieron en las siguientes herramientas:

- Glide dock xp: Es una herramienta de screening virtual y docking, creada por Schrödinger, capaz de encontrar hits de ligandos proteicos con un receptor, asistiendo en la optimización de proyectos de drug discovery basados en estructuras proteicas.[2]

- CADD ASFP: Es un servicio web basado en inteligencia artificial, desarrollado por el grupo CADD, que provee una plataforma con herramientas computacionales para permitir la caracterización de la interacción proteína-ligando.[30]

Estos datos obtenidos son valores numéricos que describen cómo se realiza la unión entre el ligando (que consiste en el sistema que se quiere usar como inhibidor) y el receptor (que es la proteasa principal del COVID-19, la M-Pro). Estos describen las posibles capacidades que se tienen para realizar la unión, como la capacidad de formar enlaces covalentes, enlaces hidrófobos, etc. Como los propios programas ya los preparan para ser empleados, al ser valores numéricos, y proporcionados mediante ficheros csv, no se ha de realizar ningún tipo de preprocesado de los datos.

Estos dos programas operan de forma similar, se introducen dos ficheros, con el mismo nombre, y finalizando en *_ligand.sdf en el caso que sea el ligando, y *_protein.pdb, en caso de que sea el receptor. Estos formatos de ficheros están especializados para el paso de datos de fragmentos de proteínas (sdf, en el caso del ligando), o para describir sistemas proteicos completos (pdb, en el caso de proteínas). Estos ficheros se guardan en carpetas, separando cada sistema, y se guardan todas en un fichero .zip. Este fichero, se introduce en cada programa, y producirá un fichero .csv con los datos de los descriptores de cada sistema.

Estos datos de entrada forman una tabla conjunta con 104 entradas (filas) y 34 columnas. En cada fila, se describe la unión ligando-receptor entre la molécula de interés y la M-Pro. Las columnas componen las 34 características que describen la interacción que se forma en cada sistema, obtenidas al juntar los descriptores calculados mediante los dos programas mencionados anteriormente.

Una vez se obtienen los descriptores de los 104 sistemas que se tienen, se usarán ambos ficheros csv creados como input para el software, juntando los datos en un único dataframe de pandas para poder tratarlos.

2.1 Creación de un modelo inicial, empleando TPOT y PLS

El primer estudio que se realiza para encontrar un modelo adecuado para nuestro sistema consistió en emplear las herramientas de TPOT y PLS, y comprobar si se pueden utilizar para alcanzar un modelo eficaz automatizado con TPOT, y aumentar su efectividad reduciendo la dimensionalidad de los datos mediante PLS.

La teoría tras esta decisión reside en la facilidad que aporta la herramienta de TPOT en el momento de optimizar un pipeline de *machine learning*, suponiendo esta una solución, rápida pero eficaz para el problema planteado, en el caso de que los resultados sean adecuados.

Para la elaboración del modelo, se extraen las características obtenidas por ambos programas empleados (Glide dock y CADD ASFP) y combinan en un único dataframe. Se dividen los sistemas contenidos en el dataframe en dos conjuntos de forma aleatoria, con una proporción de 20% (empleado como conjunto de test) y 80% (empleado como conjunto de train).

A partir de estos conjuntos, se comprueban diversos tipos de algoritmos, y se comparan mediante el empleo de diversas métricas, como el coeficiente de correlación entre el pIC50 predicho y el real (R^2), el MAE (*Mean Absolute Error)*, el MSE (*Mean Squared Error*), y el RMSE (*Root Mean Squared Error)*, como muestra el código 1.

```
    print('Score:', modelStep.score(totalDf[featuresStep], y))

    print('Mean Absolute Error:', metrics.mean_absolute_error(y_true,
y_pred))
    print('Mean Squared Error:', metrics.mean_squared_error(y_true,
y_pred))
    print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_true, y_pred)))
```

Código 1. Obtención de las métricas de evaluación.

El problema que surge tras estas primeras pruebas consiste en que empleando esta herramienta no se alcanza un rendimiento considerable.

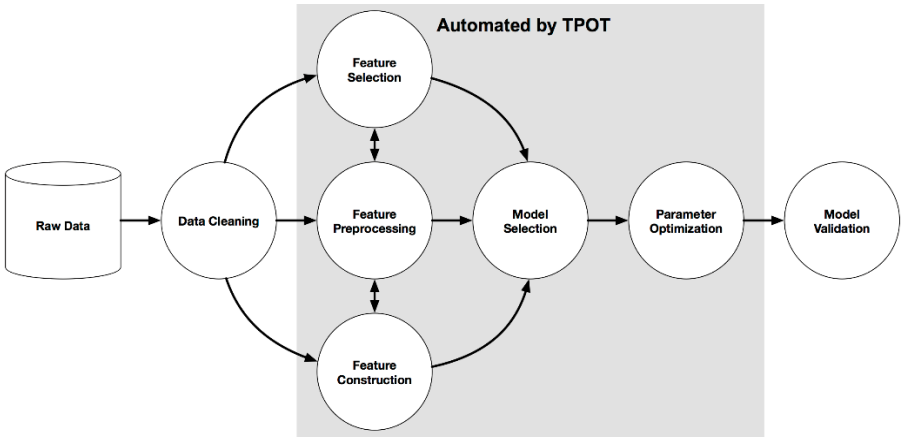

Figura 11: Automatización de modelo por TPOT.

Se considera que uno de los factores que puede afectar al rendimiento del modelo vía TPOT puede ser causado por la elevada dimensionalidad de los datos de entrada, refiriéndose a esta como el conjunto de diferentes variables que describe cada sistema.

Al emplear un elevado número de variables, la distancia media entre los datos aumenta con el número de dimensiones y la variabilidad de la distancia disminuye exponencialmente con el número de dimensiones. Esto produce, por consecuente, que los datos estén todos a casi la misma distancia, sin que haya variabilidad entre sus distancias. Esto se conoce como el problema de la dimensionalidad, y es uno de los principales causantes de malfuncionamiento en la creación de un modelo adecuado de *machine learning*.[1]

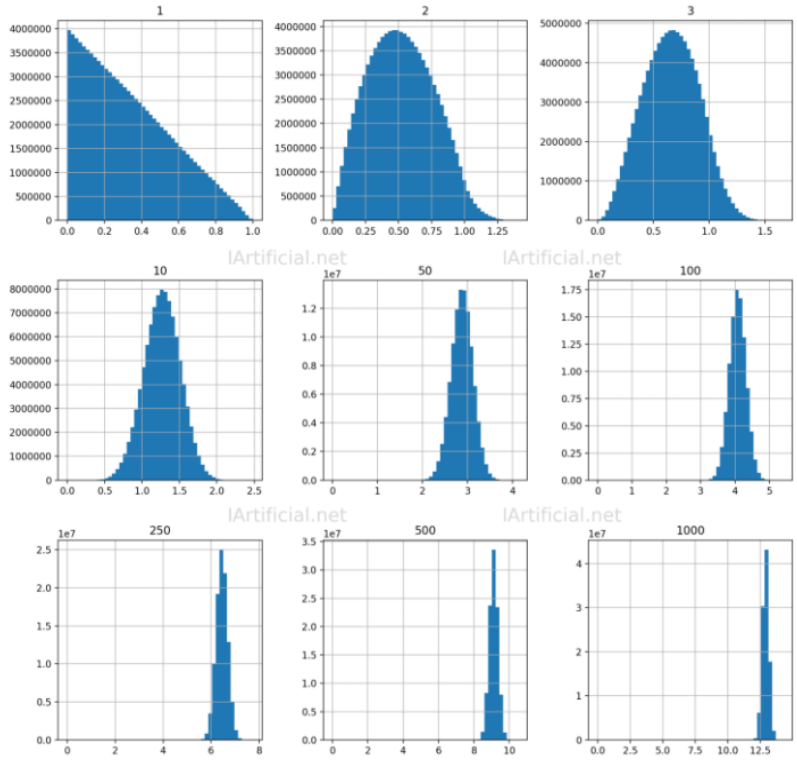

Ilustración 12: Distribución de las distancias en distintas dimensiones.[1]

Como se puede observar, conforme van aumentando las dimensiones la variabilidad va desapareciendo. Esto hace que los algoritmos de *machine learning* basados en distancias tengan problemas en el cálculo de los resultados.

Para optimizar el funcionamiento del modelo anterior, se emplea el método estadístico conocido como regresión de mínimos cuadrados parciales (PLS), con el objetivo de disminuir el número de dimensiones de los datos iniciales, resolviendo de esta forma el problema de la dimensionalidad, y hallar las características más relevantes que jueguen un papel dominante en la determinación del pIC50.

El objetivo principal del primer estudio que se realiza empleando PLS consiste en encontrar los parámetros a emplear en el algoritmo de PLS y en TPOT, para poder evaluar con que parámetros se obtienen los mejores resultados con el conjunto de datos, de forma independiente uno del otro para, posteriormente, evaluar un nuevo modelo empleando PLS y TPOT en conjunto y usando estos parámetros obtenidos en conjunto. De esta forma, se puede evaluar el número de dimensiones adecuadas con las que

trabajar obtenidas por PLS, y los mejores parámetros con los cuales se puede mejorar la eficacia de TPOT en sistema de datos utilizados.

Se comprueba, inicialmente, cuáles son los mejores parámetros para TPOT y para PLS por separado, realizando 4 pruebas para cada conjunto de parámetros y usando como valor final la media de los coeficientes de correlación, y poder reducir el número de variables con las que trabajar en el modelo que emplee TPOT y PLS en conjunto.

PLS		
Componentes	Coeficiente Regresión	Media
4	0,335216771	0,322038
	0,308111845	
	0,351726909	
	0,293097153	
8	0,439290962	0,386854
	0,400277331	
	0,304686605	
	0,403162881	
9	0,341651986	0,445609
	0,454311674	
	0,583310245	
	0,403162881	
10	0,301215135	0,367542
	0,401236425	
	0,415123541	
	0,348156235	

Tabla 1: Regresión mediante PLS.

Como se puede observar en la tabla 1, los mejores valores de PLS se obtienen para cuando se emplean 9 componentes, por lo cual, se puede estimar que al emplearlo en conjunto con TPOT, para poder lograr los mejores resultados, se emplearán 9 componentes para PLS. Si se aumenta el número de componentes por encima de 9, se comprueba que se produce una disminución de la efectividad del modelo, por lo que 9 es el mejor valor para este sistema.

TPOT			
Generaciones	Población	Coeficiente Regresion	Media
2 gen	20	0,148535552	0,33759471
		0,324725534	
		0,401357226	
		0,475760546	
4 gen	10	0,120845476	0,32188758
		0,375474585	
		0,292420563	
		0,498809709	
	20	0,601017014	0,58989618
		0,544105656	
		0,705894484	
		0,508567572	
	40	0,457306139	0,44258653
		0,352244021	
		0,590603252	
		0,573093782	
6 gen	20	0,558110998	0,44258653
		0,392740472	
		0,286743822	
		0,532750843	
	40	0,457306139	0,4458118
		0,382244021	
		0,400603252	
		0,543093782	
8 gen	20	0,336084727	0,45425191
		0,568833427	
		0,191182249	
		0,720907243	

Tabla 2: Regresión mediante TPOT.

Como se puede observar en la Tabla 2, las mejores regresiones se obtienen con los siguientes parámetros de TPOT: 4 generaciones, población 20.

Una vez determinados los mejores parámetros para cada método de forma aislada, se puede elaborar un modelo que emplee PLS para reducir las dimensiones de los datos de entrada, para que TPOT emplee esos datos para preparar el pipeline de optimización, y alcanzar mejores resultados. Al emplear estos parámetros, y ligeras variaciones de estos, se reduce el número de posibilidades a estudiar, al localizar, de forma previa, en qué valores se podrán obtener los mejores resultados. (Tabla 3).

TPOT+PLS				
Generaciones TPOT	Población TPOT	Componentes PLS	Coeficiente Regresión	
4 gen	10	4	0,601017014	0,496878743
			0,392740472	
		8	0,568833427	0,556469542
			0,544105656	
		9	0,590603252	0,496883067
			0,403162881	
	20	4	0,304686605	0,45285181
			0,601017014	
		8	0,573093782	0,581848517
			0,590603252	
		9	0,705894484	0,607231028
			0,508567572	
	40	4	0,558110998	0,54543092
			0,532750843	
		8	0,645017014	0,594561335
			0,544105656	
		9	0,400603252	0,560755248
			0,720907243	

Tabla 3: Regresión mediante TPOT+PLS.

Realizando una regresión con ambas herramientas, se obtiene que la mejor combinación de parámetros consiste en el empleo de 4 generaciones de TPOT, población igual a 20 en TPOT y 9 componentes de PLS.

Aunque los resultados iniciales son decentes al observar los valores de los coeficientes de correlación obtenidos, se opta por desechar estos modelos y preparar nuevos modelos de menor complejidad algorítmica (empleando la librería de sklearn), debido a un problema que surge durante las ejecuciones, que limita nuestro estudio.

Los modelos que emplean estos dos métodos conjuntos tienen el problema de ocupar demasiada memoria de la máquina que lo esté ejecutando, y al no poseer un computador de elevado rendimiento, se producen faltas de ausencia de espacio suficiente en el fichero de swap como para ejecutar el modelo, lo que derivó a un fin de los ensayos con este modelo.

También se probó a realizar una red neuronal simple en esta etapa de desarrollo, pero, al igual que con el modelo anterior, el computador empleado para la ejecución era incapaz de realizar cualquier modelo usando la red neuronal, por lo que también se descartó.

2.2 Comparación de diversos modelos, feature importance y stepwise regresion

Tras dejar de emplear las herramientas de PLS y TPOT, se opta estudiar nuevos modelos, empleando los propuestos en el apartado teórico, al ser modelos simples de crear, o que se adaptan de una forma adecuada al problema que se plantea, y estudiar cual de todos los modelos elaborados es capaz de describir el sistema entre receptor-ligando que se quiere estudiar, y predecir, de forma efectiva, el pIC50 de nuevos compuestos a estudiar.

Para poder analizar los modelos y resolver el problema de la dimensionalidad que se describe en el apartado anterior, se emplean dos tipos de algoritmos, conocidos como algoritmos de *feature importance*, que permite obtener un listado de las características más importantes en el momento del cálculo de la regresión, generando así tres conjuntos de pruebas a partir de los datos que se poseen:

- Prueba completa: Una prueba en la que se evalúa el conjunto completo de características para cada sistema (el conjunto de todos los descriptores obtenidos), sin tratar estas características.

- Prueba con p-value < 0.05: Una prueba en la cual se evalúa un conjunto de características (descriptores) que presenten un valor p inferior a 0.05 al realizar una regresión lineal de Pearson entre la característica y el valor final a predecir (pIC50). Se establece este máximo para el p-value al ser considerado, desde un punto de vista estadístico, que los p-values inferiores a 0.05 son estadísticamente significativos.

  Este algoritmo, explicado en el apartado 1.3.2, sigue el siguiente fragmento de código (código 1), en el cual, totalDf es el dataframe de pandas en el cual se guardan los datos conjuntos de ambas herramientas para obtener los descriptores, y featuresTotal es una lista con todas las características que se estudian, y, mediante el empleo de la regresión lineal de Pearson, se pueden determinar las características significativas.

```
y = totalDf['pIC50']
listGoodPvalue = []
for column in featuresTotal:
    x = totalDf[column]
    r, p = scipy.stats.pearsonr(x, y)
    if (p<=0.05): listGoodPvalue.append(column)
```

  Código 1. Obtención de características con p-valor inferior a 0.05.

- Prueba con *Stepwise regresion*: Una prueba en la cual se evalúa un conjunto de características que son obtenidas al realizar una regresión de tipo *Stepwise*. Mediante este algoritmo, podemos obtener una lista de características que son estadísticamente significativas para el cálculo del pIC50, sabiendo que estas características son linealmente independientes unas de otras.

  El algoritmo empleado para realizar esta regresión es el siguiente:

```
def forward_regression(X, y,
                       threshold_in,
                       verbose=False):
    initial_list = []
    included = list(initial_list)
    while True:
        changed=False
        excluded = list(set(X.columns)-set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y,
sm.add_constant(pd.DataFrame(X[included+[new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed=True
            if verbose:
                print('Add  {:30} with p-value
{:.6}'.format(best_feature, best_pval))

        if not changed:
            break

    return included
```

Código 2. Forward regresion, Stepwise regresion.

Empleando la regresión lineal de Pearson entre las características y el pIC50, se obtiene una lista de variables que tienen correlación significativa con la variable final a determinar, de forma que se puede reducir también el número de variables con las que se opera. Como resultado se obtienen las siguientes variables:

['XP LowMW', 'XP RotPenal', 'XP Sitemap', 'crit5', 'crit6_continue', 'lig_vol', 'pock_vol', 'charge_score'].

En la regresión *Stepwise* se obtiene un conjunto independiente de variables que influencian, de forma significativa, al cálculo de la variable dependiente en un modelo, por lo que permitirá reducir de forma significativa el conjunto de variables con las que se trabaja, y el posible ruido que generen.

La diferencia respecto al conjunto anterior es que todas las variables son independientes entre sí, mientras que las obtenidas mediante la regresión lineal, pueden ser dependientes unas con otras, lo cual explica que se obtenga una lista de tamaño inferior.

Esta regresión, como resultado, nos devuelve una lista de 4 variables que sabemos, de forma garantizada, que resultan críticas en el cálculo del pIC50:

['XP RotPenal', 'lig_vol', 'crit5', 'PP-dst'].

Se puede extraer que la regresión *Stepwise* ha sido exitosa al comprobar que la mayor parte de las variables obtenidas por esta se presentan en las variables obtenidas por la regresión lineal de Pearson.

Para los estudios que se realizarán con estos conjuntos para determinar el algoritmo que se empleará, no se usará la estrategia de dividir el conjunto de datos en training y test, sino que se valorará, para realizar tanto el training como el testing del modelo, el conjunto de datos en su totalidad. Se toma esta decisión para poder evitar la posible variación de resultados que se producen entre diferentes ejecuciones, y se emplea, para poder validar los resultados, una validación cruzada: *leave-one-out cross-validation*. De esta forma, se pueden realizar estudios que son reproducibles mientras que se garantiza que los datos obtenidos son válidos.

Se prueban los siguientes tipos de modelos de regresión: regresión lineal, regresión de árbol de decisión, regresión de procesos gaussianos, regresión PLS (empleando 4, 8, 12 y 16 componentes respectivamente), regresión contraída con R (*ridge regresion*), regresión polinomial (empleando un polinomio de grado 2, al disponer únicamente de 107 moléculas de prueba y aproximadamente una veintena de variables) y una regresión de cresta bayesiana (*Bayesian Ridge Regression*).

Para cada modelo se realizará, a su vez, un tipo de validación cruzada, la *leave-one-out cross-validation*, mencionada anteriormente, mediante la cual podemos comparar todos los modelos a estudiar, empleando todos los datos disponibles para la obtención de un error estimado sin tener variabilidad entre los resultados mientras se garantiza una permitiendo así, comparar los modelos con datos constantes y reproducibles, complementando, así, los datos obtenidos por las regresiones con este valor.

		Lineal Regression	**Decision Tree Regression**	**Gaussian Process Regressor**	**PLSRegression (nº comp = 4)**	**PLS Regression (nº comp = 8)**	**PLS Regression (nº comp = 12)**	**PLS Regression (nº comp = 16)**	**Ridge Regression**	**Polinomial Regression**	**Bayesian Ridge Regression**
Empleando todas las características	Score (R^2):	0.5153	1.0	0.4311	0.4026	0.4624	0.4849	0.4913	0.4106	1.0	0.2303
	Mean Absolute Error:	0.3882	1,30E-07	0.4278	0.4295	0.4172	0.3992	0.3949	0.4332	1,29E-07	0.4905
	Mean Squared Error:	0.2257	2,12E-14	0.2649	0.2782	0.2503	0.2398	0.2369	0.2744	2,12E-14	0.3584
	Root Mean Squared Error:	0.4751	1,46E-07	0.5147	0.5274	0.5003	0.4897	0.4867	0.5239	1,46E-07	0.5987
Valor de leave one out cross validation	MSE	0.525	0.577	0.490	0.460	0.460	0.460	0.460	0.472	Crashes	0.422
	Desviación estandar	0.782	0.739	0.668	0.610	0.610	0.610	0.610	0.609	Crashes	0.509

Tabla 4: Prueba de diferentes modelos empleando todas las características de los sistemas.

		Lineal Regression	**Decision Tree Regression**	**Gaussian Process Regressor**	**PLSRegression (nº comp = 4)**	**PLS Regression (nº comp = 8)**	**PLS Regression (nº comp = 12)**	**PLS Regression (nº comp = 16)**	**Ridge Regression**	**Polinomial Regression**	**Bayesian Ridge Regression**
Empleando solo las características con p-value < 0.05	Score (R^2):	0.2636	1.0	0.2607	0.2601	0.2636	0.2636	0.2636	0.2584	0.2619	0.2154
	Mean Absolute Error:	0.4670	0.0	0.4679	0.4706	0.4670	0.4670	0.4670	0.4691	0.4742	0.4899
	Mean Squared Error:	0.3429	0.0	0.3443	0.3446	0.3429	0.3429	0.3429	0.3454	0.3437	0.3654
	Root Mean Squared Error:	0.58561	0.0	0.5868	0.5871	0.5856	0.5856	0.5856	0.5877	0.5863	0.6045
Valor de leave one out cross validation	MSE	0.415	0.589	0.413	0.410	0.410	0.410	0.410	0.412	Crashes	0.398
	Desviación estandar	0.544	0.818	0.540	0.537	0.537	0.537	0.537	0.539	Crashes	0.502

Tabla 5: Prueba de diferentes modelos empleando las características con un p-value < 0.05 en los sistemas.

		Lineal Regression	**Decision Tree Regression**	**Gaussian Process Regressor**	**PLSRegression (nº comp = 4)**	**PLS Regression (nº comp = 8)**	**PLS Regression (nº comp = 12)**	**PLS Regression (nº comp = 16)**	**Ridge Regression**	**Polinomial Regression**	**Bayesian Ridge Regression**
Empleando las características de Stepwise regression	Score (R^2):	0.2872	1.0	0.2843	0.2872	0.2872	0.2872	0.2872	0.2758	0.3539	0.2089
	Mean Absolute Error:	0.4746	0.0	0.4726	0.4746	0.4746	0.4746	0.4746	0.4724	0.4504	0.4884
	Mean Squared Error:	0.3319	0.0	0.3333	0.3319	0.3319	0.3319	0.3319	0.3373	0.3009	0.3684
	Root Mean Squared Error:	0.5761	0.0	0.5773	0.5761	0.5761	0.5761	0.5761	0.5807	0.5485	0.6069
Valor de leave one out cross validation	MSE	0.374	0.730	0.374	0.374	0.374	0.374	0.374	0.376	Crashes	0.389
	Desviación estandar	0.441	1.031	0.448	0.441	0.441	0.441	0.441	0.457	Crashes	0.480

Tabla 6: Prueba de diferentes modelos empleando las características obtenidas por una Stepwise regresión en los sistemas.

Tomando los resultados obtenidos tras aplicar la *Leave-one-out cross-validation* y ordenarlos de menor a mayor MSE, se puede apreciar que, a excepción de la regresión de árbol de decisión, se obtienen mejores resultados en el resto de los casos después de filtrar las variables por aquellas con un p-value inferior a 0.05, e incluso mejores resultados aún con las variables seleccionadas por la *stepwise regression.*

La regresión de árbol produce unos resultados extremadamente positivos, pero que son debidos a un caso de *overfitting*, el cual se produce ya que esta regresión ha creado una "hoja" del árbol de regresión por cada caso, lo que produce valores muy positivos para casos del conjunto de datos usado como training, pero disminuye la efectividad de forma elevada con datos diferentes. Esto lleva al descarte de este modelo, por lo que se consideran, únicamente, los valores obtenidos por la *cross-validation*.

Las diferencias en términos de MSE entre los mejores modelos son mínimas, encontrándose 6 modelos con valores igualados de MSE, por lo que se opta por tomar el modelo más simple de estos mejores modelos, que corresponde al modelo de regresión lineal (tabla 7).

Model	**Variables**	**MSE**	**SD**
Lineal Regression	stepwise	0,374	0,441
PLS Regression (nº comp = 4)	stepwise	0,374	0,441
PLS Regression (nº comp = 8)	stepwise	0,374	0,441
PLS Regression (nº comp = 12)	stepwise	0,374	0,441
PLS Regression (nº comp = 16)	stepwise	0,374	0,441
Gaussian Process Regressor	stepwise	0,374	0,448
Ridge Regression	stepwise	0,376	0,457
Bayesian Ridge Regression	stepwise	0,389	0,480
Bayesian Ridge Regression	p-value	0,398	0,502
PLS Regression (nº comp = 4)	p-value	0,410	0,537
PLS Regression (nº comp = 8)	p-value	0,410	0,537
PLS Regression (nº comp = 12)	p-value	0,410	0,537
PLS Regression (nº comp = 16)	p-value	0,410	0,537
Ridge Regression	p-value	0,412	0,539
Gaussian Process Regressor	p-value	0,413	0,540
Lineal Regression	p-value	0,415	0,544
Bayesian Ridge Regression	all	0,422	0,509
PLS Regression (nº comp = 4)	all	0,460	0,610

PLS Regression (nº comp = 8)	all	0,460	0,610
PLS Regression (nº comp = 12)	all	0,460	0,610
PLS Regression (nº comp = 16)	all	0,460	0,610
Ridge Regression	all	0,472	0,609
Gaussian Process Regressor	all	0,490	0,668
Lineal Regression	all	0,525	0,782
Decision Tree Regression	all	0,577	0,739
Decision Tree Regression	p-value	0,589	0,818
Decision Tree Regression	stepwise	0,730	1,031

Tabla 7: Comparación del MSE de los modelos probados.

Tras decidir emplear una regresión lineal, se procede a analizar los parámetros que se obtienen al emplearla, como los coeficientes que poseen las cuatro características empleadas por la *Stepwise regresion* en la regresión lineal. Para esto, se emplea del módulo de statmodel la regresión OLS, al ser también una regresión lineal, también sirve para poder estudiar el sistema. OLS nos proporciona una tabla en la que aparecen detalladas diversas características del modelo, y los coeficientes de regresión de la regresión lineal empleadas, entre otros valores.

3 Análisis de los resultados

Los modelos por los que se optan finalmente son el modelo OLS y el modelo de regresión lineal simple, al emplear ambos el mismo algoritmo para calcular la regresión (empleando regresiones lineales).

Al emplear el modelo obtenido mediante la regresión OLS, se pueden emplear el conjunto de herramientas de análisis que nos proporciona esta regresión. Con ellas, se obtiene una tabla en la que se puede visualizar los coeficientes de regresión lineal, el coeficiente de correlación, el número de variables empleadas… variables que muestran la calidad de los resultados de la regresión.

OLS Regression Results

Dep. Variable:	pIC50	R-squared (uncentered):	0.972
Model:	OLS	Adj. R-squared (uncentered):	0.971
Method:	Least Squares	F-statistic:	892.4
Date:	Tue, 11 May 2021	Prob (F-statistic):	5.93e-79
Time:	10:27:36	Log-Likelihood:	-135.79
No. Observations:	107	AIC:	279.6
Df Residuals:	103	BIC:	290.3
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
XP RotPenal	-0.0015	0.001	-1.586	0.116	-0.003	0.000
lig_vol	0.0127	0.002	7.819	0.000	0.009	0.016
crit5	1.6342	0.543	3.008	0.003	0.557	2.712
PP-dst	0.4993	0.173	2.893	0.005	0.157	0.841

Omnibus:	0.299	Durbin-Watson:	1.532
Prob(Omnibus):	0.861	Jarque-Bera (JB):	0.053
Skew:	-0.012	Prob(JB):	0.974
Kurtosis:	3.106	Cond. No.	2.57e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 2.57e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Figura 13: Análisis del modelo lineal mediante OLS.

Por otro lado, se analiza el modelo de regresión lineal simple empleando las herramientas de métricas de sklearn, mostradas en el siguiente fragmento de código:

```
    print('Mean Absolute Error:', metrics.mean_absolute_error(y_true,
CrossVal_Pred))
    print('Mean Squared Error:', metrics.mean_squared_error(y_true,
CrossVal_Pred))
    print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_true, CrossVal_Pred)))
    r = np.corrcoef(y_true, CrossVal_Pred)
    print('Score (R^2 value):', r[0,1])
```

Código 3. Obtención de las métricas de evaluación de la regresión lineal.

De esta forma, se obtienen siguientes resultados correspondientes a la regresión lineal simple:

- MAE: 0.50
- MSE: 0.374
- RMSE: 0.611
- R-Squared: 0.454

Para poder comparar los resultados entre ambas regresiones, se realiza también, empleando Excel, una regresión lineal de las variables seleccionadas por la *Stepwise regresion* frente al valor del pIC50, sin especificar, para el caso de OLS, una intercepción con el eje de coordenadas (al ser así la regresión que obtiene OLS), y una regresión en la que si haya una intercepción con el eje de coordenadas, que corresponde a la regresión lineal simple. Se emplea Excel para poder obtener un mayor detalle en las características que describan las regresiones obtenidas. Estas dos regresiones estadísticas se muestran en las tablas 8 y 9.

Para determinar cuál de estos dos modelos representa mejor al sistema, se compararán empleando, de forma principal, el error estándar que presenten, al haber sido el determinante para elegir el modelo con el que continuar de los estudiados, y en caso de similitud, se usará el coeficiente de correlación.

En el momento de comparar los valores obtenidos por las regresiones en ambos modelos, se puede observar, que se obtienen grandes diferencias entre un modelo y otro, principalmente debidas a que la regresión OLS no emplea término independiente en la hora del empleo de la regresión lineal, mientras que el modelo de la regresión lineal simple calcula un término independiente en la regresión.

Esta diferencia entre ambos modelos genera diferencias entre los resultados de la predicción de un modelo y otro, observándose, en las tablas 8 y 9, que la regresión OLS presenta un coeficiente de correlación muy elevado (0’986), pero también se produce un aumento del error estándar (0’878).

El modelo de regresión lineal simple presenta un coeficiente de correlación inferior a OLS (0’536), pero a su vez, presenta un error inferior (0’59), por lo cual, se presentará, en este ámbito, como mejor modelo.

Respecto al *p-value* de ambos modelos, ambos presentan valores muy próximos a cero, inferiores a 0.05, por lo que, respecto a esta variable, resultan aceptables. Esto reafirma que los modelos obtenidos son significativos, incluso teniendo en cuenta que se ha trabajado con una muestra de tamaño reducido, contando solo con 107 muestras.

Teniendo en cuenta estos valores, se determina, finalmente, que el modelo de regresión lineal simple es el mejor modelo de los estudiados, al presentar el menor *mean squared error* entre los modelos estudiados, y que produce unos valores estadísticamente aceptables.

Para complementar este estudio, se realizan unos estudios mostrando la correlación obtenida de forma gráfica, empleando el módulo de *pyplot*, de *matplotlib*.

Se realiza una gráfica para estudiar los resultados, en la cual se compara el pIC50 predicho con el real. Si el modelo obtenido representa de forma adecuada los datos, esta gráfica tendrá un aspecto lineal. Cuanto más se acerquen los puntos a una correlación lineal, más preciso será el modelo.

Se puede comprobar que los datos presentan una correlación lineal con bastante precisión, teniendo en cuenta los pocos datos que se poseen para realizar el entrenamiento y el testeo del modelo.

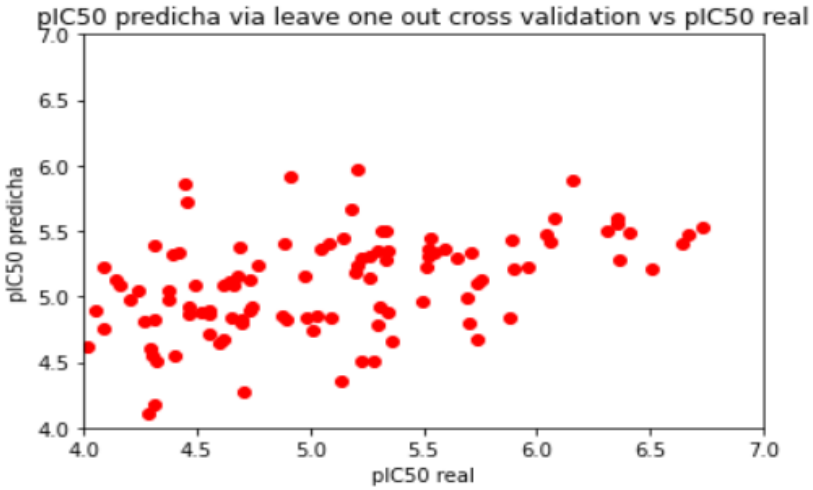

Figura 13: pIC50 predicho frente al pIC50 real.

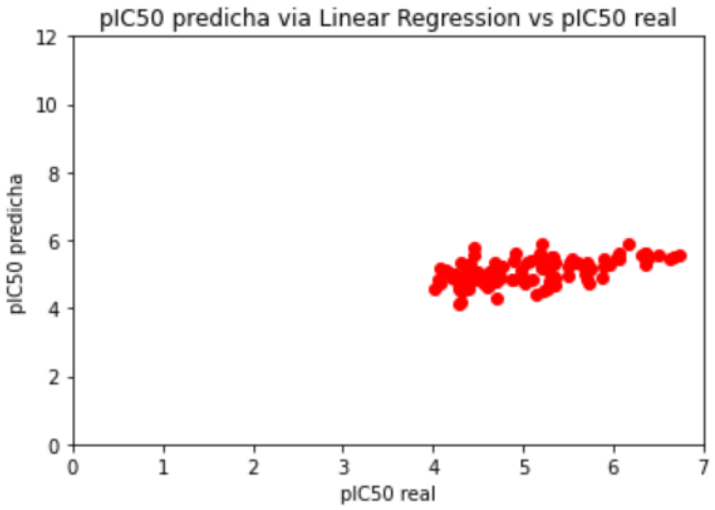

Figura 14: Visualización del modelo de regresión lineal simple.

Como se puede comprobar en estas visualizaciones, los datos con los que se trabaja se acercan a una aproximación lineal, por lo que el modelo lineal, dentro de todos los modelos que se han estudiado, es el que mejor se aproxima a los datos con lo que se trabaja, permitiendo obtener un modelo capaz de predecir de forma aceptable el pIC50 de los sistemas inhibidor-proteasa del COVID-19.

Regresión lineal simple

Estadísticas de la regresión	
Coeficiente de correlación múltiple	0,535968939
Coeficiente de determinación R^2	0,287262704
R^2 ajustado	0,259312221
Error típico	0,590133007
Observaciones	107

ANÁLISIS DE VARIANZA

	Grados de libertad	*Suma de cuadrados*	*Promedio de los cuadrados*	*F*	*Valor crítico de F*
Regresión	4	14,31692474	3,579231186	10,27755806	4,9423E-07
Residuos	102	35,52221051	0,348256966		
Total	106	49,83913526			

	Coeficientes	*Error típico*	*Estadístico t*	*Probabilidad*	*Inferior 95%*	*Superior 95%*	*Inferior 95.0%*	*Superior 95.0%*
Intercepción	5,977609275	0,533188679	11,2110581	1,7241E-19	4,92003209	7,03518646	4,92003209	7,035186462
XP								
RotPenal	-0,002266357	0,000659477	-3,43660015	0,00085359	-0,00357443	-0,00095829	-0,00357443	-0,000958289
lig_vol	0,004431945	0,001320889	3,35527346	0,00111445	0,00181197	0,00705192	0,00181197	0,007051923
crit5	-1,522262918	0,461280667	-3,3000796	0,00133225	-2,43721091	-0,60731493	-2,43721091	-0,607314927
PP-dst	-0,299943253	0,136198987	-2,2022429	0,02990015	-0,57009329	-0,02979321	-0,57009329	-0,029793215

Tabla 8: Resultados regresión lineal simple.

Regresión lineal mediante OLS

Estadísticas de la regresión	
Coeficiente de correlación múltiple	0,985877511
Coeficiente de determinación R^2	0,971954467
R^2 ajustado	0,961428869
Error típico	0,877407207
Observaciones	107

ANÁLISIS DE VARIANZA

	Grados de libertad	*Suma de cuadrados*	*Promedio de los cuadrados*	*F*	*Valor crítico de F*
Regresión	4	2748,032357	687,0080893	892,3997829	2,1627E-78
Residuos	103	79,29387092	0,769843407		
Total	107	2827,326228			

	Coeficientes	*Error típico*	*Estadístico t*	*Probabilidad*	*Inferior 95%*	*Superior 95%*	*Inferior 95.0%*	*Superior 95.0%*
Intercepción	0	#N/D	#N/D	#N/D	#N/D	#N/D	#N/D	#N/D
XP RotPenal	-0,001548203	0,00097587	-1,586484454	0,115694812	-0,003483612	0,000387206	-0,003483612	0,000387206
lig_vol	0,012722741	0,00162725	7,818552594	4,79942E-12	0,009495474	0,015950007	0,009495474	0,015950007
crit5	1,634191309	0,54326142	3,008112205	0,003304164	0,556760442	2,711622175	0,556760442	2,711622175
PP-dst	0,499259108	0,172547312	2,893462091	0,004650622	0,157052219	0,841465996	0,157052219	0,841465996

Tabla 9: Resultados regresión OLS.

4 Conclusiones

Gracias a haber participado en este estudio he podido conocer un poco del mundo de *machine learning* y conocer sus metodologías, los elementos que se emplean en estos estudios, aprender sobre el proceso de elaboración y optimización de un pipeline de *machine learning* y el tratamiento de los resultados y conocer un poco sobre los procesos de *deep learning*.

El modelo final obtenido, pese a no poseer un coeficiente de correlación muy elevado, es capaz de predecir con bajo error el pIC50 que presentan diversos compuestos tras introducir los datos que describan un sistema molecular, obtenidos mediante *glide_dock_XP* y CADD_ASFP, dos programas disponibles en el ámbito biotecnológico.

Uno de los motivos principales que causan este bajo coeficiente de correlación se debe al limitado tamaño de muestras que se poseen para este estudio, consistiendo en un total de 107 sistemas de muestras, lo que provoca una falta de entrenamiento para que el modelo llegue a presentar coeficientes de correlación elevados.

Pese a este inconveniente, tras estudiar un conjunto de diferentes modelos, se obtiene un modelo con un MSE muy reducido, capaz de establecer una correlación con un p-valor próximo a cero, que indica que el modelo es estadísticamente aceptable, y obteniendo gráficas en las que se relacionan los valores predichos con los reales que muestran, claramente, una correlación lineal. El valor R que se obtiene finalmente es de 0’536, siendo un valor R significativo para un tamaño de muestra pequeño, manteniendo un MSE reducido, de 0,59, por lo que se puede concluir que el modelo final obtenido resulta adecuado para futuros estudios destinados al descubrimiento de nuevos inhibidores para la proteasa principal del COVID-19.

Al haber empleado la técnica de *leave-one-out cross-validation* se ha podido descartar la posibilidad de haber causado *overfitting* en el estudio de los diferentes modelos, por lo que los datos empleados en la comparación son válidos para poder estudiar qué modelo representa mejor los datos con los que se trabaja.

Se pretende más adelante eliminar una de las cuatro variables obtenidas mediante la *stepwise regression*, el *'XP RotPenal'*, ya que para obtenerla se ha de utilizar un programa de alto coste computacional y de licencia de pago, por lo que encontrar un modelo resulte eficiente y preciso calculando el pIC50 sin esa variable sería muy efectivo para poder desplegar el modelo a una plataforma online y que se encuentre disponible para la comunidad científica. Al obtener 7 variables empleando la regresión de Pearson entre las variables independientes y la variable dependiente, y 4 mediante la *stepwise regresion*, al eliminar esta variable, otra de las 3 variables descartadas puede ‘reemplazarla’, al poder haber sido descartada por tener correlación con esta variable.

Otro estudio que se puede realizar consistiría en seguir la línea de estudio empleada inicialmente, con las técnicas de TPOT y PLS, usando un computador con elevada capacidad que permita utilizar ambas técnicas sin quedarse sin espacio en la memoria de paginación, permitiendo continuar sin problemas dicha línea, o, en su defecto, estudiar el sistema empleando la regresión *Stepwise* junto a TPOT.

Además, también se podrán realizar pruebas con un conjunto de muestras mayor y entrenar y verificar la eficacia del modelo con menor dependencia del número de muestras, logrando así, eliminar uno de los principales problemas que se han tenido.

Además, se podrán probar otro tipo de modelos para compararlos con el modelo de regresión lineal y comprobar si hay otro que pueda emplearse. Se ha considerado emplear, una vez se aumente el número de muestras, emplear el modelo de *Random Forest*, al ser un modelo particularmente fuerte en el caso de que haya presencia de ruido en los datos, pudiendo realizar cálculos incluso en el caso de la falta de alguna variable, aunque se evaluarán otros modelos también.[3]

Finalmente, he aprendido como se puede incorporar la ingeniería informática y los sistemas de *machine learning* a las investigaciones biotecnológicas para poder agilizar los estudios y poder facilitar la computación de nuevos resultados, por lo que la inclusión de equipos de trabajo informáticos en el proceso de investigación de nuevos fármacos puede resultar crucial para poder descubrir nuevas medicinas con una alta tasa de efectividad sin tener que realizar estudios en laboratorio sin una dirección pre-definida.

Con esto, considero que la carrera y, sobre todo, este trabajo de fin de grado, me han hecho crecer como persona a la vez que me han formado para poder adquirir nuevas competencias, esenciales para un programador, que me permitirán afrontar nuevos retos que surjan en un futuro.

5 Autoevaluación

Lo primero que quiero resaltar en este apartado consiste en un sentimiento, que solo ha aumentado durante la realización de estas prácticas y este trabajo de fin de grado, el haber podido realizar un proyecto, personal, dirigido por mí, al cual me haya podido enfrentar con mis conocimientos y alcanzar unos resultados exitosos, con la ayuda de mis tutores. Me ha permitido investigar sobre métodos para unir las dos carreras que he cursado, para encontrar esa unión que no quedaba del todo clara en mis estudios, y poder usar esos conocimientos para lograr un fin, que pueda ser utilizado en la investigación.

Este trabajo no me ha dejado indiferente, pues ha generado por una confianza en mí, a la vez que me ha ayudado a identificar y detectar algunas de mis carencias y necesidades formativas.

Se han cumplido muchas expectativas, formando parte de un grupo de investigación biotecnológico, los cuales han compartido sus conocimientos conmigo, permitiéndome conocer cómo se trabaja en el ámbito bioinformático y de desarrollo de software con funcionalidad en el ámbito de la investigación, facilitándome así, adquirir nuevas habilidades y destrezas que me pueden servir en mi futuro laboral, lo que me hace sentir lleno de satisfacción y de orgullo.

Pienso que la aportación que he logrado realizar, descrita en este trabajo de 45 páginas, puede llegar a suponer un avance y una nueva rama de herramientas para la investigación de nuevos fármacos contra el COVID-19, empleando herramientas especializadas en el ámbito de *machine learning*, las cuales se han visto complementadas por los conocimientos proporcionados por mis profesores durante estos 5 años. Esta experiencia me ha permitido, en conjunto, mejorar mi formación experimental, así como reafirmar mi vocación por mis estudios realizados.

Con esto, considero que las carreras y, sobre todo, este trabajo de fin de grado, me han hecho crecer como persona a la vez que me han formado para poder adquirir nuevas competencias, esenciales para un programador y para un biotecnólogo, que me permitirán afrontar nuevos retos que surjan en un futuro.

Referencias

[1] "La Maldición de la Dimensión en Machine Learning - IArtificial.net." https://www.iartificial.net/la-maldicion-de-la-dimension-en-machine-learning/ (accessed Jun. 08, 2021).

[2] "Docking and Scoring | Schrödinger." https://www.schrodinger.com/science-articles/docking-and-scoring (accessed Jun. 07, 2021).

[3] "Gradient Boosting vs Random Forest | by Abolfazl Ravanshad | Medium." https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80 (accessed Jun. 07, 2021).

[4] "Stepwise Regression." https://www.investopedia.com/terms/s/stepwise-regression.asp (accessed Jun. 07, 2021).

[5] "TPOT." http://epistasislab.github.io/tpot/ (accessed Jun. 07, 2021).

[6] R. S. Olson and J. H. Moore, "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning," 2019, pp. 151–160.

[7] "TPOT - AutoML." http://automl.info/tpot/ (accessed Jun. 07, 2021).

[8] R. S. Olson and J. H. Moore, "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning," Springer, Cham, 2019, pp. 151–160.

[9] "Regresión de mínimos cuadrados parciales - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Regresión_de_mínimos_cuadrados_parciales (accessed Jun. 07, 2021).

[10] "Modelo de Regresión PLS," 2017.

[11] "¿Qué es la regresión de mínimos cuadrados parciales? - Minitab." https://support.minitab.com/es-mx/minitab/18/help-and-how-to/modeling-statistics/regression/supporting-topics/partial-least-squares-regression/what-is-partial-least-squares-regression/ (accessed Jun. 07, 2021).

[12] "What are Neural Networks? | IBM." https://www.ibm.com/cloud/learn/neural-networks (accessed Jun. 07, 2021).

[13] "Ridge Regression | Brilliant Math & Science Wiki." https://brilliant.org/wiki/ridge-regression/ (accessed Jun. 07, 2021).

[14] "Multicolinealidad - Qué es, definición y concepto | 2021 | Economipedia." https://economipedia.com/definiciones/multicolinealidad.html (accessed Jun. 07, 2021).

[15] M. Seeger, "Gaussian processes for machine learning.," *International journal of neural systems*, vol. 14, no. 2. pp. 69–106, 2004, doi: 10.1142/S0129065704001899.

[16] "Gaussian process - Wikipedia." https://en.wikipedia.org/wiki/Gaussian_process (accessed Jun. 07, 2021).

[17] C. E. Rasmussen, "Gaussian Processes in machine learning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3176, pp. 63–71, 2004, doi: 10.1007/978-3-540-28650-9_4.

[18] "Getting started with Gaussian process regression modeling | by Boyang Zhao | Towards Data Science." https://towardsdatascience.com/getting-started-with-gaussian-process-regression-modeling-47e7982b534d (accessed Jun. 07, 2021).

[19] E. Pekel, "Estimation of soil moisture using decision tree regression," *Theor. Appl. Climatol.*, vol. 139, no. 3–4, pp. 1111–1119, Feb. 2020, doi: 10.1007/s00704-019-03048-8.

[20] "Polynomial Regression. This is my third blog in the Machine… | by Animesh Agarwal | Towards Data Science." https://towardsdatascience.com/polynomial-regression-bbe8b9d97491 (accessed Jun. 07, 2021).

[21] "Machine learning Polynomial Regression - Javatpoint." https://www.javatpoint.com/machine-learning-polynomial-regression (accessed Jun. 07, 2021).

[22] K.-Y. Lee *et al.*, "Comparison and Analysis of Linear Regression & Artificial Neural Network," 2017. Accessed: Jun. 07, 2021. [Online]. Available: http://www.ripublication.com.

[23] S. Rong and B.-W. Zhang, "The research of regression model in machine learning field," doi: 10.1051/matecconf/201817601033.

[24] S. Kavitha, S. Varuna, and R. Ramya, "A comparative analysis on linear regression and support vector regression," May 2017, doi: 10.1109/GET.2016.7916627.

[25] "SciPy.org — SciPy.org." https://www.scipy.org/ (accessed Jun. 07, 2021).

[26] "Matplotlib: Python plotting — Matplotlib 3.4.2 documentation." https://matplotlib.org/ (accessed Jun. 07, 2021).

[27] "Scikit-Learn, herramienta básica para el Data Science en Python." https://www.master-data-scientist.com/scikit-learn-data-science/ (accessed Jun. 07, 2021).

[28] “scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/ (accessed Jun. 07, 2021).
[29] “Anaconda | The World’s Most Popular Data Science Platform.” https://www.anaconda.com/ (accessed Jun. 07, 2021).
[30] “ASFP.” http://cadd.zju.edu.cn/asfp/ (accessed Jun. 07, 2021).