Universitat Oberta de Catalunya

uoc.edu

Predicción de la toxicidad en péptidos mediante técnicas Machine Learning

Mariano Monserrat Gómez
Máster Universitario en Bioinformática y Bioestadística
Bioinformática Farmacéutica

Tutor: Melchor Sánchez Martínez
Profesor responsable de la asignatura: Marc Maceira Duch

05/01/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Predicción de la toxicidad en péptidos mediante técnicas Machine Learning*
Nombre del autor:	*Mariano Monserrat Gómez*
Nombre del consultor/a:	*Melchor Sánchez Martínez*
Nombre del PRA:	*Marc Maceira Duch*
Fecha de entrega:	01/2021
Titulación:	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Bioinformática Farmacéutica*
Idioma del trabajo:	*Castellano*
Palabras clave	*Péptido, Toxicidad, Machine Learning*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Durante el desarrollo de nuevos fármacos, se ha visto que la mayoría de las moléculas estudiadas son descartadas en los ensayos clínicos por presencia de toxicidad. A consecuencia, se han desarrollado métodos computacionales para predecir la actividad de las moléculas candidatas. En los últimos años, los péptidos se han propuesto como posibles candidatos a fármacos por su alta actividad biológica, especificidad, bajo coste de producción y alta penetración.

En este trabajo se pretende desarrollar un modelo Machine Learning (ML) para predecir la toxicidad peptídica para el posible desarrollo de nuevos fármacos. Se recopilarán péptidos tóxicos y no tóxicos de distintas bases de datos y se agruparán en un dataset. Con las secuencias peptídicas, se generarán los descriptores pseudo-aminoácidos para la creación de modelos de predicción de toxicidad peptídica. Con métodos de clustering y balanceo se crearán nuevos datasets que serán usados para generar modelos predictivos con los algoritmos de ML utilizados: SVM, RF y GBRT. Se seleccionarán los mejores modelos y serán evaluados con un dataset externo.

Los modelos generados por clustering y balanceo eran de mayor calidad (exactitud, precisión, sensibilidad, especificidad, F1 y AUC de curva ROC) que los obtenidos con el dataset inicial. De los cinco mejores modelos evaluados con el dataset externo (Modelo Subsampling SVM, Modelo Subsampling RF, Modelo DBSCAN+Subsampling SVM, Modelo DBSCAN+Subsampling GBRT y Modelo Linclust RF), tres presentaban mejores indicadores de calidad.

Se concluye que los mejores modelos para predecir la toxicidad de péptidos son Modelo Subsampling SVM, Modelo Subsampling RF y Modelo DBSCAN+Subsampling GBRT.

Abstract (in English, 250 words or less):

In the development of new drugs, it has been seen that the majority of the studied molecules are discarded during clinical trials due to toxicity. As a consequence, computational methods have been developed to predict the activity of candidate molecules. In the last years, peptides have been proposed as possible drug candidates due to their high biological activity, specificity, low production cost and high penetration. This project aims to develop a Machine Learning (ML) model to predict peptide toxicity for the possible development of new drugs. Toxic and non-toxic peptides will be collected from different databases and gathered in a new dataset. With the peptide sequences, the pseudo-amino acids descriptors will be generated for the creation of predictive models of peptide toxicity. New datasets will be created with clustering and balancing techniques that will be used to generate predictive models with the used ML algorithms: SVM, RF and GBRT. The best models will be selected and evaluated with an external dataset.

The models generated by clustering and subsampling had higher quality (accuracy, precision, sensitivity, specificity, F1-scrore and AUC of the ROC curve) than those obtained with the initial dataset. Three of the five best models evaluated with an external dataset (Modelo Subsampling SVM, Modelo Subsampling RF, Modelo DBSCAN+Subsampling SVM, Modelo DBSCAN+Subsampling GBRT y Modelo Linclust RF) presented better quality indicators.

We conclude that the best models to predict peptides toxicity are ‘Modelo Subsampling SVM’, ‘Modelo Subsampling RF’ and ‘Modelo DBSCAN+Subsampling GBRT’.

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

El mercado farmacéutico se está viendo obligado a evolucionar como consecuencia de la gran presión económica que está sufriendo. Por un lado, las autoridades públicas están presionando para reducir los costes de la atención médica mediante la disminución de los precios de los medicamentos y la sustitución por genéricos [1]. Por otro lado, las autoridades reguladoras son más exigentes en términos de eficacia, calidad y seguridad de los medicamentos, lo que, combinado con el progreso biotecnológico, ha provocado un aumento de los costes farmacéuticos de I+D [1].

Actualmente, el 38% de los fármacos candidatos en desarrollo se abandonan en la fase I de los ensayos clínicos debido a la toxicidad que presentan, el 63% de los que llegan a la fase II de los ensayos clínicos se retiran por falta de eficacia o mala biodisponibilidad, el 45% de los fármacos restantes fracasan en los ensayos clínicos de fase III y el 23% de los que cumplen los ensayos clínicos no están autorizados por la Administración de Alimentos y Medicamentos (FDA) o la Agencia Europea de Medicamentos (EMEA) [2]. El abandono por falta de eficacia, toxicología y seguridad clínica son las causas más frecuentes de fracaso en el desarrollo de nuevas moléculas. Generalmente, la tasa de fracaso durante el desarrollo de fármacos es del 90% [2,3].

Como consecuencia, las empresas farmacéuticas deben mejorar las estrategias para el desarrollo de fármacos con el objetivo de seguir siendo competitivas, rentables y por tanto, ser capaces de reducir sus costes [4]. También, deben ser innovadoras para descubrir nuevos fármacos y comercializar productos innovadores, así como dar una segunda vida a sus moléculas con el fin de buscar nuevas aplicaciones terapéuticas [5].

En este ámbito, los péptidos juegan un papel importante como posibles candidatos a fármacos, ya que tienen numerosas ventajas sobre las moléculas pequeñas que incluyen alta actividad biológica, alta especificidad, bajo coste de producción y alta penetración [6,7, 8].

Sin embargo, el uso de péptidos en el desarrollo de fármacos también presenta los siguientes inconvenientes:

- Toxicidad.
- Inmunogenicidad.
- Estabilidad.

La estabilidad de los péptidos se puede mejorar mediante la incorporación de D- aminoácidos, el cambio de la estructura química de la molécula y la ciclación [9]. De igual modo, existen métodos 'in silico' capaces de predecir la inmunogenicidad de los péptidos. Sin embargo, los métodos para predecir la toxicidad de los péptidos están todavía en desarrollo [10]. Los métodos computacionales en general, y en particular los modelos de Machine Learning, pueden ayudar a paliar este problema de clasificación de la toxicidad peptídica [8]. Por tanto, predecir la toxicidad de los péptidos no solo ahorraría tiempo y dinero, sino que también facilitaría el diseño de mejores péptidos terapéuticos conservando sus funcionalidades y la baja toxicidad [8, 10].

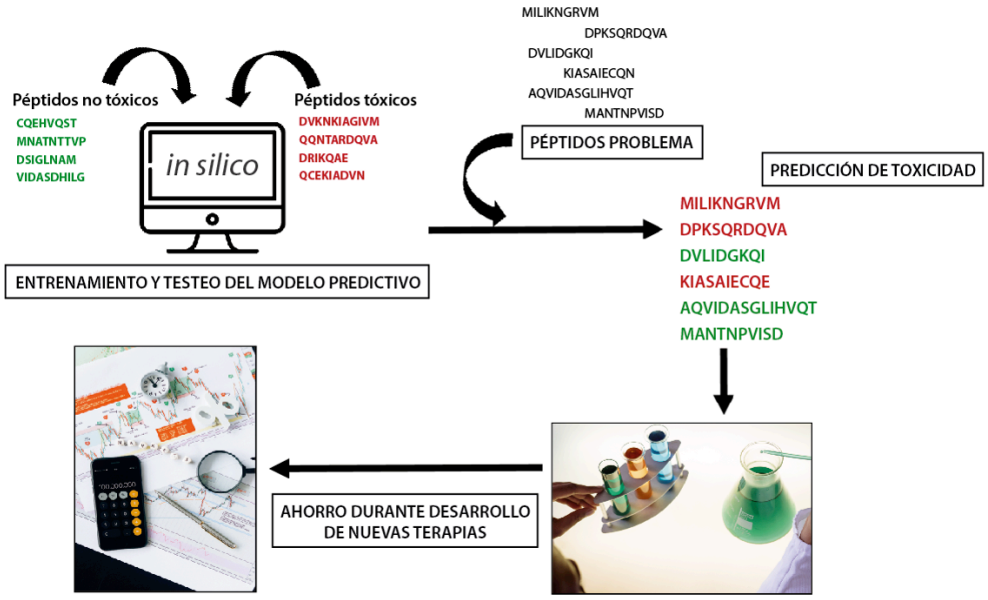


Fig. 1. Planteamiento gráfico del proyecto.

En este proyecto se desea desarrollar un método para predecir la toxicidad de los péptidos. Se recopilarán péptidos tóxicos y no tóxicos de distintas bases de datos y se utilizarán para entrenar modelos Machine Learning con el objetivo de, dado un péptido nuevo, predecir su toxicidad.

1.2. Objetivos del Trabajo

El objetivo del trabajo es la creación de un modelo predictivo capaz de clasificar de forma fiable a los péptidos como tóxicos o no tóxicos mediante técnicas de Machine Learning. De esta forma, se busca crear un nuevo método que permita ahorrar tiempo y dinero, además de facilitar el diseño de mejores péptidos terapéuticos sin perder sus funcionalidades. Por ello, como objetivos principales, tendríamos la creación de una base de datos de péptidos tóxicos y no tóxicos y la generación de los distintos modelos predictivos mediante técnicas de Machine Learning.

Estos dos objetivos principales se podrían desglosar en los siguientes objetivos específicos:

1) Analizar las distintas bases de datos de péptidos conocidas.
2) Crear una base de datos de péptidos tóxicos y no tóxicos.
3) Obtener las características cuantitativas de cada secuencia peptídica de la base de datos creada.
4) Seleccionar las distintas técnicas de Machine Learning destacables en bioinformática.
5) Generar los modelos predictivos a partir de las técnicas ML seleccionadas.
6) Analizar cuantitativamente mediante métricas de calidad los modelos creados para ver qué modelos se ajustan mejor a los datos.
7) Evaluar la calidad de predicción de los mejores modelos generados a partir de un conjunto de datos externo de péptidos.

1.3. Enfoque y método seguido

Este Trabajo se ha enfocado con el objetivo de generar un modelo capaz de clasificar los péptidos como tóxicos o no tóxicos mediante el uso de distintas técnicas de Machine Learning. Basándonos en el modelo creado, se busca, dado un conjunto de péptidos externo, predecir de forma correcta y fiable su toxicidad.

Comenzaremos creando un fichero con todas las secuencias de péptidos obtenidas a partir de cada base de datos analizada durante la etapa de revisión bibliográfica. Partiendo de las secuencias peptídicas, obtendremos las características cuantitativas y generaremos los distintos modelos predictivos, que serán evaluados mediante distintas métricas de calidad.

Para realizar este estudio, se ha hecho uso del lenguaje de programación Python, ya que es de gran utilidad en múltiples aplicaciones dentro del campo de la bioinformática [11]. Este lenguaje se ha revolucionado con la generalización del Big Data, Inteligencia Artificial, Machine Learning, Deep Learning y el surgimiento de 'data science', ya que la mayoría de este tipo de herramientas se han desarrollado en este lenguaje [12]. Para nuestro estudio, Python, presenta librerías como 'numpy' o 'pandas' que usaremos para tratar grandes conjuntos de datos; o la librería 'scikit-learn' en el área de Machine Learning que usaremos para la generación de los modelos predictivos.

Principalmente, haremos uso del paquete iFeature de Python para la extracción y selección de características de secuencias peptídicas [13]. Haremos uso de sus funciones para obtener las características cuantitativas de cada secuencia peptídica, agrupar y buscar de outliers o seleccionar las características cuantitativas más importantes.

1.4. Planificación del Trabajo

Los recursos utilizados para realizar este trabajo son las distintas bases de datos de donde hemos obtenido los péptidos. Para tratar cada una de las bases de datos, hemos creado un script en Python para leer las secuencias peptídicas que contiene cada base de datos y añadir a cada péptido un identificador y su correspondiente etiqueta de tóxico y no tóxico. Finalmente, en el mismo script, hemos creado un conjunto de datos donde se han agrupado todos los péptidos y se han eliminado los duplicados. Este conjunto de datos creado es el recurso principal para la realización del proyecto.

Todos los scripts utilizados se encuentran en la plataforma GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML). Para ejecutar los scripts, es necesario descargar los datasets usados. Se pueden encontrar en el siguiente enlace de Google Drive: TFM-Peptides-Prediction-ML.

En la Fig. 2., se muestran las tareas realizadas.

Nombre	Fecha de inicio	Fecha de fin
PEC0 – Definición de los contenidos del trabajo	16/9/20	28/9/20
PEC1 – Plan de trabajo	29/9/20	13/10/20
PEC2 – Desarrollo del trabajo – Fase 1	14/10/20	16/11/20
PEC3 – Desarrollo del trabajo – Fase 2	17/11/20	14/12/20
PEC4 – Cierre de la memoria	15/12/20	5/1/21
PEC5a – Elaboración de la presentación	6/1/21	8/1/21
PEC5b – Defensa pública	13/1/21	20/1/21
Bases de datos	14/10/20	27/10/20
Análisis de BBDD de péptidos conocidas	14/10/20	19/10/20
Crear base de datos de péptidos tóxicos y no tó...	19/10/20	27/10/20
Obtener las características cuantitativas de cada ...	19/10/20	27/10/20
Generar el dataset de péptidos	27/10/20	27/10/20
Estudiar técnicas de Machine Learning	27/10/20	30/10/20
Seleccionar los algoritmos Machine Learning	27/10/20	30/10/20
Modelos predictivos	30/10/20	1/12/20
Generar los modelos predictivos	30/10/20	9/11/20
Entrenar los modelos	9/11/20	24/11/20
Testar los modelos predictivos	16/11/20	1/12/20
Obtención del algoritmo de predicción de toxicidad	1/12/20	1/12/20
Análisis de los modelos generados	1/12/20	14/12/20
Métricas de calidad	1/12/20	14/12/20
Evaluar la calidad de predicción a partir de pépti...	4/12/20	14/12/20
Escribir memoria TFM	26/10/20	5/1/21
Generar memoria y presentación TFM	8/1/21	8/1/21
Defensa pública	20/1/21	20/1/21

Fig. 2. Tareas realizadas en el proyecto.

A continuación, mostramos la planificación temporal de cada tarea mediante un diagrama de Gantt.

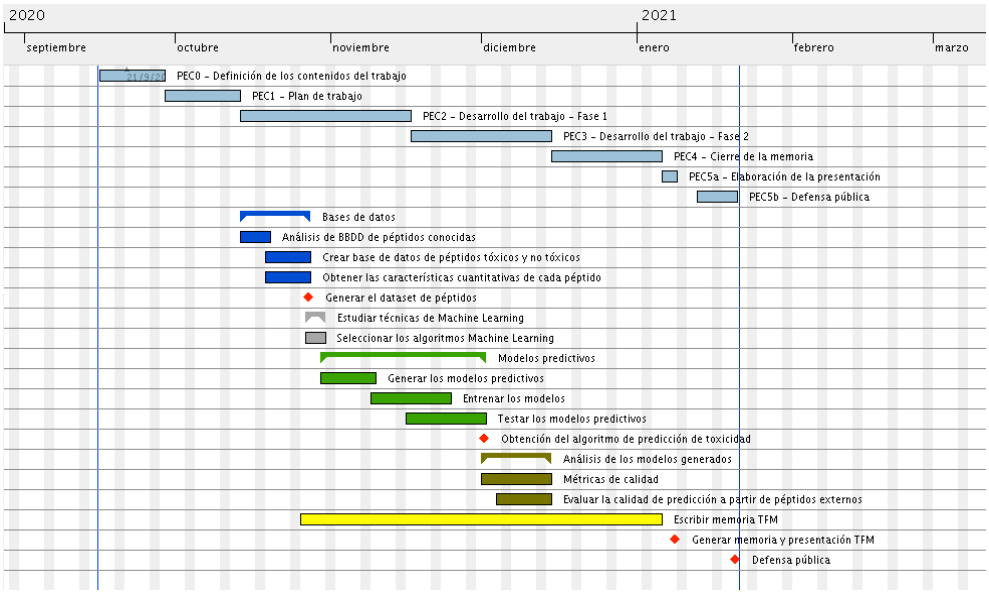


Fig. 3. Diagrama de Gantt, planificación temporal.

Los hitos, que se encuentran representados en el diagrama de Gantt como rombos en color rojo son los siguientes:

1. Generar el dataset de péptidos.
2. Obtención del algoritmo de predicción de toxicidad.
3. Generar la memoria y presentación del TFM.
4. Defensa pública del TFM.

Tal y como comentamos en la 'PEC. 2. Desarrollo del trabajo - Fase 1.', uno de los riesgos que presenta el proyecto, es la obtención de unos resultados poco fiables y de baja calidad debido a un desequilibrio en el conjunto de datos creado de péptidos tóxicos y no tóxicos.

Este problema no ha provocado un retraso temporal ya que desde un primer momento se tuvo en cuenta como un análisis de riesgo. Por ello, se estableció un espacio temporal muy amplio en la tarea 'Generar modelos predictivos'.

Como análisis de riesgo, se pensaron las siguientes soluciones:

· Balancear el conjunto de datos.
· Clustering y detección de outliers.

Todas las técnicas utilizadas se explicarán detalladamente en el apartado '2. Materiales y Métodos'.

1.5. Breve sumario de productos obtenidos

Por un lado, a partir de las distintas bases de datos de péptidos estudiadas, hemos obtenido un dataset con un total de 480656 péptidos (11364 tóxicos y 469292 no tóxicos). Dataset creado en el script 'BBDD.py' denominado 'All_peptides.txt'. Este dataset contiene un identificador (ID), la secuencia peptídica y la etiqueta (Label) de toxicidad para cada péptido.

A partir de este dataset se han obtenido las características cuantitativas, en concreto, de los descriptores pseudo-aminoácidos (PAAC) de cada péptido y se ha creado el dataset 'encodings_Allpeptides.tsv'.

Por otro lado, se ha hecho una selección de las mejores características cuantitativas mediante la técnica χ^2 de iFeature del dataset 'All_peptides.txt'. Generando un dataset con la mejor característica y otro con las dos mejores.

Para cada dataset creado hasta el momento, hemos realizado tres modelos predictivos variando el tipo de algoritmo utilizado (Support Vector Machine, Random Forest y Gradient Boasted Tree). Estos modelos creados se pueden encontrar en la plataforma GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML), en los scripts de Python 'Algoritmo SVM.py', 'Algoritmo RF.py' y 'Algoritmo GBRT.py', respectivamente.

Los modelos, para cada algoritmo, obtenidos como productos son:

→ Algoritmo SVM:

· '**Modelo mejor descriptor'**: se selecciona el mejor descriptor PAAC resultante de utilizar la estrategia χ^2 en el dataset 'All_peptides.txt' con todos los descriptores pseudo-aminoácidos y se utiliza para generar este modelo.

→ Algoritmo RF:

· '**Modelo dataset inicial**': usa el dataset 'All_peptides.txt' para generar el modelo predictivo.

· '**Modelo mejor descriptor'**: utiliza un dataset con el mejor descriptor PAAC resultante de utilizar la estrategia χ^2 para el dataset 'All_peptides.txt' con todos los descriptores PAAC.

· '**Modelo 2 mejores descriptores**: se utiliza técnica χ^2 para seleccionar los dos mejores descriptores del dataset 'All_peptides.txt' con todos los descriptores PAAC y generar un modelo RF.

→ Algoritmo GBRT (hemos obtenido los mismos modelos que para el caso del algoritmo RF):

· '**Modelo dataset inicial**': usa el dataset 'All_peptides.txt' para generar el modelo predictivo.

· '**Modelo mejor descriptor'**: utiliza un dataset con el mejor descriptor PAAC resultante de utilizar la estrategia χ^2 para el dataset 'All_peptides.txt' con todos los descriptores PAAC.

· '**Modelo 2 mejores descriptores**: se utiliza técnica χ^2 para seleccionar los dos mejores descriptores del dataset 'All_peptides.txt' con todos los descriptores PAAC y generar un modelo RF.

Los resultados para los casos comentados se han guardado dentro de la carpeta 'Resultados Modelos' disponible en GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML).

Tras obtener los resultados de los modelos realizados anteriormente y analizarlos, se han realizado distintas técnicas para tratar el dataset inicial 'All_peptides.txt':

- Balanceo del dataset inicial utilizando la técnica subsampling. En este caso, se ha generado un dataset balanceado y, a partir de éste, se han generado los modelos predictivos.
- Se ha creado un nuevo dataset realizando un clustering DBSCAN partir de las características cuantitativas de cada péptido del dataset de 480656 péptidos. Con el dataset originado, entrenamos y testeamos distintos modelos de predicción.
- A partir del dataset anterior, después de realizar el clustering DBSCAN, se balancean las categorías mediante la estrategia subsampling de la clase mayoritaria. El dataset resultante se utiliza para generar modelos predictivos.
- Por último, realizamos un clustering por secuencia del dataset inicial de 480656 péptidos para generar modelos predictivos.

Con estos datasets creados, también se han realizado modelos predictivos con los algoritmos Support Vector Machine, Random Forest y Gradient Boasted Tree. Para cada algoritmo utilizado, se han obtenido los siguientes 4 modelos predictivos:

· '**Modelo Subsampling**': usa el dataset resultante de balancear 'All_peptides.txt' siguiendo la estrategia Subsampling.

· '**Modelo DBSCAN**': utiliza el dataset resultante de realizar un clustering a partir de los descriptores PAAC del dataset 'All_peptides.txt'.

· '**Modelo DBSCAN+Subsampling**': se balancea siguiendo la estrategia Subsampling el dataset utilizado en el modelo DBSCAN anterior.

· '**Modelo Linclust**': usa el dataset resultante de realizar un clustering por secuencias del dataset 'All_peptides.txt'.

Estos modelos están disponibles en los scripts 'Algoritmo SVM.py', 'Algoritmo RF.py' y 'Algoritmo GBRT.py', almacenados en GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML).

Finalmente, tras evaluar los resultados de cada modelo generado anteriormente, se han seleccionado los mejores modelos predictivos para probarlos con un dataset de péptidos externo. Este nuevo dataset, denominado 'dataset_ejemplo.csv', se puede localizar en la carpeta 'PEPTIDOS EXTERNOS' del proyecto.

Se ha creado el script 'dt_pepExternos.py' que clasifica los péptidos del dataset 'dataset_ejemplo.csv' a partir de los mejores modelos seleccionados.

En este caso, se han obtenido los siguientes modelos:

- '**Modelo Subsampling SVM**'.
- '**Modelo Subsampling RF**'.
- '**Modelo DBSCAN+SUBSAMPLING SVM**'.
- '**Modelo DBSCAN+Subsampling GBRT**'.
- '**Modelo Linclust RF**'.

Todos los productos y resultados obtenidos están almacenados en el proyecto 'peptides prediction - ML' que he subido a la plataforma GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML).

Adicionalmente, se puede acceder al proyecto completo con todos los scripts, datasets utilizados, ficheros obtenidos, etc., a través de la plataforma Google Drive: TFM-Peptides-Prediction-ML.

1.6. Breve descripción de los otros capítulos de la memoria

En este apartado, vamos a realizar una breve introducción de los siguientes apartados que vamos a tratar.

2. Materiales y Métodos: en este caso, primero se comenta cómo hemos obtenido el dataset inicial del que partimos en este proyecto. Seguidamente, se explican las características cuantitativas utilizadas para generar los modelos ML. Además, explicamos las distintas técnicas que hemos utilizado para tratar el dataset inicial y así, crear nuevos conjuntos de datos con los que crear otros modelos predictivos.

Por otro lado, se han detallado los software principales que se han utilizado en el proyecto. Se han explicado los algoritmos Machine Learning usados y se han comentado las métricas de calidad que tendremos en cuenta para evaluar los modelos creados y obtener conclusiones.

3. Resultados y Discusión: en este apartado, se comentan los parámetros seleccionados según el tipo de modelo utilizado, para posteriormente, entrenar y testear los modelos. Se muestran los resultados obtenidos para cada modelo según el tipo de algoritmo Machine Learning utilizado, y se analizan para ver cuáles son los que tienen una mayor calidad. Por último, se realiza una selección de los mejores modelos predictivos, y a partir de un dataset de péptidos externo, se vuelven a evaluar, pero esta vez realizando predicciones sobre este nuevo dataset.

4. Conclusiones: teniendo en cuenta los resultados del apartado ‘3. Resultados y Discusión’, en esta parte del proyecto se analizan las distintas conclusiones que se obtienen para este proyecto.

2. Materiales y Métodos

2.1. Creación del dataset

Los datos de este estudio se han obtenido a partir de las siguientes bases de datos: ToxinPred [14], ArchnoServer [15], Swiss-Prot (usando para los péptidos tóxicos la palabra clave 'KW-0800' [16] y para los péptidos no tóxicos 'NOT KW-0800' [17]), celiacDataBase [18], NTXpred [19], BTXpred [20], Kalium [21], T3DB [22] y DBETH [23]. De cada base de datos hemos recopilado la secuencia de cada péptido y hemos añadido una etiqueta dependiendo de si el péptido es tóxico o no. Finalmente, hemos creado un dataset con todas las secuencias de péptidos agrupadas con su correspondiente etiqueta y un identificador, obteniendo, una vez eliminados los duplicados, un total de 480656 péptidos (11364 tóxicos y 469292 no tóxicos).

2.2. Desarrollo de los modelos predictivos

2.2.1. Creación de los datasets utilizados en los modelos predictivos

2.2.1.1. Descriptores pseudo-aminoácidos (PAAC)

La composición de los pseudo-aminoácidos fue introducida originalmente por Kuo-Chen Chou en 2001 para predecir atributos celulares de proteínas en función de sus secuencias de aminoácidos con el objetivo de mejorar la predicción de la localización subcelular de proteínas y la asociación con la bicapa lipídica de un orgánulo [24].

Es una combinación de un conjunto de factores de correlación de secuencias discretas y de los 20 componentes de la composición de los aminoácidos convencionales [24]. El conjunto de factores de correlación es un valor promedio de las tres propiedades de los aminoácidos: valor de hidrofobicidad, valor de hidrofilicidad y masa de la cadena lateral.

Por lo que, para cada secuencia proteica, los pseudo-aminoácidos generan 20 + λ números discretos. La idea de la composición de los pseudo-aminoácidos es, por un lado, incluir la característica principal de la composición de los aminoácidos que refleja la aparición de uno de los 20 aminoácidos en una proteína, y por otro lado, incluir información más allá de dicha composición a través de los factores de correlación [25].

Por lo tanto, se ha observado que mediante su uso, existe una mejora notable en la calidad de la predicción [24], ya que, como se ha comentado, contiene más información a parte de la composición de los aminoácidos y por la tanto, puede reflejar mejor las características de una secuencia de proteína a través de un modelo discreto [24].

En este proyecto, he utilizado la función disponible en el software iFeature para obtener los descriptores PAAC de cada secuencia peptídica (función almacenada en GitHub como 'funcion_PAAC.py'). Esta función recibe como entrada un fichero en formato fasta que contiene las secuencias peptídicas y devuelve el fichero con todos los descriptores PAAC de cada péptido.

La creación del dataset con estos descriptores es la base de nuestro proyecto, ya que es el dataset utilizado para generar los primeros modelos predictivos. Además, es el dataset tratado mediante técnicas de balanceo y clustering para obtener nuevos datasets a partir de los cuales realizar nuevos modelos predictivos.

2.2.1.2 Prueba chi-cuadrado (χ^2)

En estadística descriptiva, este método se aplica para probar la independencia entre dos variables. Esta prueba fue desarrollada en el año 1900 por Karl Pearson [26].

χ^2 es una medida que indica cuánto se desvían entre sí las frecuencias esperadas E y las frecuencias observadas N [13]. Un valor alto de χ^2 indica que la hipótesis de independencia, que implica que los conteos esperados y observados son similares, es incorrecta. Esta puntuación se puede utilizar para seleccionar las n características con los valores más altos según el estadístico chi-cuadrado. Esta prueba debe contener características no negativas como booleanos o frecuencias [13].

$$\chi^2 = \Sigma\ (A - E)^2 / E$$

La prueba χ^2 se ha utilizado para la selección de los mejores descriptores PAAC usando la función 'funcion_chiSQUARE.py' proporcionada por el software iFeature [13]. Mediante esta estrategia, se seleccionan los dos mejores descriptores para crear dos datasets, uno con el mejor descriptor, y el otro con los dos mejores descriptores. A partir de estos dos conjuntos de datos, se generan modelos predictivos.

2.2.1.3. Técnicas de balanceo y clustering.

Tal y como se ha comentado en el apartado anterior, se ha decido trata el dataset 'All_peptides.txt' que contiene los descriptores PAAC con distintas técnicas de balanceo y clustering.

En el capítulo 'Resultados y Discusión' comentaremos más detalladamente las causas por las que se ha decidido tratar el dataset de péptidos con las técnicas que vamos a comentar.

2.2.1.3.1. Subsampling en la clase mayoritaria

El tener un conjunto de datos con una proporción mucho mayor de una clase que otra provoca un desbalanceo en los datos que hace que los resultados obtenidos en cada modelo predictivo sean poco fiables y de baja calidad. Este tipo de problemas suele ocurrir con frecuencia en el área de la Salud [27]

La estrategia subsampling de la clase mayoritaria resuelve el problema de clasificación de clases desequilibradas. Se utiliza un algoritmo similar a K-nearest neighbor para ir eliminando observaciones de la clase mayoritaria, y así reducir la clase mayoritaria hasta igualar la clase minoritaria [27].

En este caso, se ha hecho uso de la librería Imblearn de Python para realizar el balanceo siguiendo la estrategia Subsampling.

2.2.1.3.2. Técnicas de clustering y búsqueda de outliers.

2.2.1.3.2.1. Algoritmo DBSCAN

Es un algoritmo de agrupamiento basado en la densidad (número de puntos en un radio específico), que modela los clústeres como cúmulos de alta densidad de puntos. El componente central del algoritmo DBSCAN es el concepto de muestras centrales, que son muestras que se encuentran en áreas de alta densidad. Por lo tanto, un clúster estará compuesto por un conjunto de muestras centrales, cada una cercana entre sí, y un conjunto de muestras no centrales que estarán cerca de las muestras centrales [13].

Es un algoritmo muy útil para la detección de outliers, ya que considerará como “agrupados” todos aquellos puntos de las zonas más densas y considerará como anormales aquellos puntos alejados y en zonas menos densas (normalmente valores atípicos) [28].

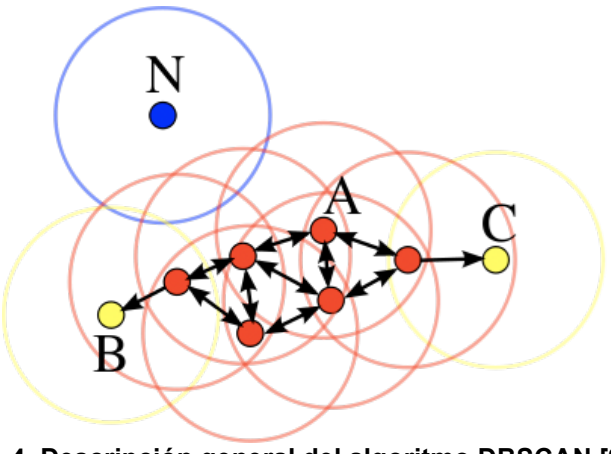


Fig. 4. Descripción general del algoritmo DBSCAN [28].

En este proyecto, se ha modificado la función DBSCAN que proporciona iFeature [13] para el agrupamiento y detección de outliers. Esta función recibe como entrada el dataset de péptidos 'All_peptides' con las características PAAC y en formato fasta.

2.2.1.3.2.2. Algoritmo Linclust/MMseqs2

Este algoritmo agrupa de forma predeterminada las entradas de un archivo FASTA mediante un algoritmo de agrupación en cascada [29]. Dispone de las opciones 'easy-cluster' y 'easy-linclust', esta última recomendada para grandes conjuntos de datos [29].

Este algoritmo posee tres criterios principales para llevar a cabo el agrupamiento entre dos secuencias:

1. Un umbral de valor E máximo que se calcula de acuerdo con las estadísticas de Karling-Altschul [31].

2. Una cobertura mínima:

- Cobertura bidireccional ("--cov-mode 0"): se define por el número de pares de residuos alineados dividido por el máximo de la longitud de las secuencias de consulta/centro y objetivo/no central [30]. Solo se agrupan las secuencias que tienen una superposición de longitud de secuencia mayor en un X% (cobertura mínima) de las más largas de las dos secuencias [32].
- Cobertura objetivo ("--cov-mode 1"): se define por el número de pares de residuos alineados dividido por la longitud de la secuencia objetivo/no central [33]. Solo se agrupan las secuencias que tienen una superposición de longitud de secuencia superior al X% (cobertura mínima) de la secuencia objetivo [32].
- Cobertura de consulta ("--cov-mode 2"): se define por el número de pares de residuos alineados dividido por la longitud de la consulta/centro [30]. En este modo, se agrupan las secuencias que tienen una superposición de longitud de secuencia superior al X% (cobertura mínima) de la secuencia de consulta [32].
- Cobertura modo 3: calcula el número de residuos alineados idénticos dividido por el número de columnas alineadas, incluyendo las columnas que contienen espacios en cualquier secuencia [32].

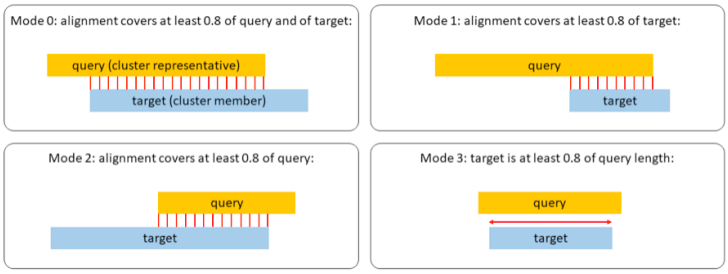


Fig. 5. Modos de cobertura [32].

3. Una identidad de secuencia mínima (--min-seq-id [0,1]): es el número de coincidencias entre residuos cuando se alinean dos secuencias [32].

En este trabajo, para el clustering por secuencia, se ha utilizado el software MMseqs2 con la opción 'easy-linclust', una cobertura de 0.6 y una identidad del 100%. Hemos hecho uso de este software con la ayuda del terminal, tal y como muestra la siguiente imagen:

```
mmseqs easy-linclust examples/All_peptides.fasta clusterRes tmp --min-seq-id 1.0 -c 0.6 --cov-mode 1
```

Fig. 6. Comando utilizado para realizar el clustering por secuencias Linclust/MMseqs2.

Los pasos generales que sigue el algoritmo Linclust de MMseqs2 para el agrupamiento de las secuencias peptídicas se pueden observar en la 'Fig.5. Descripción general del algoritmo Linclust/MMseqs2'.

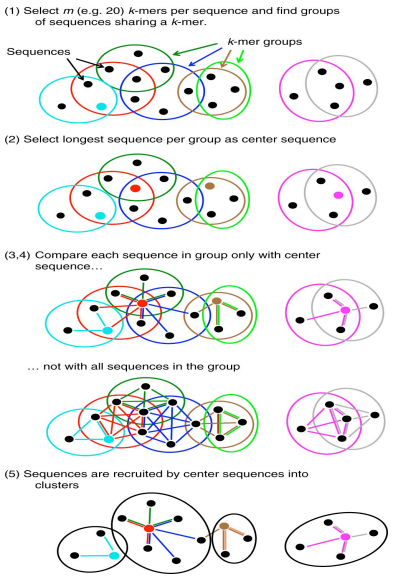


Fig. 7. Descripción general del algoritmo Linclust/MMseqs2 [30].

2.2. Datasets para entrenar los modelos

Tal y como se ha comentado en apartados anteriores, partimos de un dataset compuesto por 480656 péptidos (11364 péptidos tóxicos y 469292 no tóxicos). Para este dataset, obtenemos los descriptores pseudo-aminoácidos (PAAC) y lo empleamos para entrenar los primeros modelos generados.

1. Dataset inicial con todos los descriptores PAAC.

A partir del dataset, utilizamos la técnica χ^2 para seleccionar los mejores descriptores PAAC y crear dos nuevos datasets:

2. Dataset con el mejor descriptor PAAC.
3. Dataset con los dos mejores descriptores PAAC.

Los 3 datasets son utilizados para generar modelos y evaluar los resultados de clasificación que obtiene cada uno de ellos.

Seguidamente, tras evaluar los resultados de los primeros modelos, utilizamos técnicas de balanceo y clustering para tratar el dataset inicial (dataset compuesto por 480656 péptidos) y así, generar los siguientes datasets que también son usados para entrenar los modelos predictivos:

4. Dataset balanceado siguiendo la estrategia Subsampling de eliminación de observaciones de la clase mayoritaria.
5. Dataset generado tras realizar un clustering DBSCAN del dataset inicial que contiene los descriptores PAAC.
6. Dataset resultante del clustering DBSCAN para agrupar los péptidos según sus características PAAC y eliminar posibles outliers y balanceado siguiendo la estrategia Subsampling.
7. Dataset resultante del clustering por secuencias peptídicas usando el algoritmo Linclust/MMseqs2.

Por último, se crea un dataset externo con péptidos que no están incluidos en el dataset inicial. Se usa para valorar la calidad de clasificación de los mejores modelos predictivos generados con los datasets anteriores.

2.3. Software

2.3.1. Python

Para la realización del trabajo se ha utilizado el lenguaje de programación Python [33]. Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica [34]. Es muy útil para el desarrollo de aplicaciones, así como para la creación de scripts. Utiliza una sintaxis simple y fácil de aprender, ayudando a la legibilidad y reduciendo el coste del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código [34]. Hemos utilizado el entorno de desarrollo integrado y multiplataforma de código abierto Spyder para la programación en el lenguaje Python [35].

Python permite el uso de múltiples librerías. En este proyecto hemos hecho uso, principalmente, de las librerías Pandas para el tratamiento de datos, Imblearn para realizar el balanceo entre clases de un dataset y Sklearn para generar los modelos Machine Learning o generar las métricas de calidad para evaluar los modelos o para barajar un conjunto de datos.

2.3.2. iFeature

iFeature se compone de 'Python toolkit' y un servidor web para calcular una amplia gama de descriptores de características estructurales y fisicoquímicas a partir de secuencias proteicas o peptídicas [13].

iFeature es capaz de calcular y extraer 53 tipos diferentes de descriptores, y permite extraer propiedades específicas de aminoácidos de la base de datos AAindex [36]. Además, iFeature integra 12 tipos de algoritmos de agrupación, selección y reducción de dimensionalidad, lo que facilita la generación de características, el análisis, la capacitación y la evaluación comparativa de modelos y predicciones [36].

Para el presente trabajo, tal y como indicamos en los apartados anteriores, haremos uso de una serie de funciones presentes en 'Python toolkit', modificándolas en función de los ficheros de entrada que tenemos para nuestro proyecto. Se han utilizado funciones para obtener los descriptores pseudo-aminoácidos (PAAC), seleccionar los mejores descriptores PAAC mediante χ^2 y agrupar y detectar outliers a partir de los descriptores PAAC de cada secuencia mediante el algoritmo DBSCAN. Las funciones comentadas se pueden localizar en los scripts 'funcion_PAAC.py', 'funcion_chiSQUARE.py', 'funcion_DBSCAN.py' de GitHub (https://github.com/monserratg60/TFM-Peptides-Prediction-ML), respectivamente.

2.3.3. MMseqs2

Es un paquete software capaz de buscar y agrupar enormes conjuntos de secuencias de proteínas o nucleótidos. Es un software de código abierto con licencia GPL implementado en C++ para Linux, MacOS y Windows [37].

2.3.4. GitHub

Es una plataforma de alojamiento de código para el control de versiones y la colaboración [38]. Es de libre acceso y permite subir proyectos o scripts de código abierto o de forma privada [39].

Almacenaremos en el repositorio 'https://github.com/monserratg60/TFM-Peptides-Prediction-ML' los scripts realizados y los resultados generados en el proyecto.

Comentar que las carpetas 'data' y 'codes' son necesarias para el correcto funcionamiento de las funciones de iFeature que he utilizado en el proyecto.

Adicionalmente, se puede acceder al proyecto completo con todos los scripts, datasets utilizados, ficheros obtenidos, etc., a través de la plataforma Google Drive: TFM-Peptides-Prediction-ML

2.4. Algoritmos Machine Learning

2.4.1. Support Vector Machine (SVM)

Es un algoritmo de aprendizaje supervisado que se puede emplear para la clasificación, regresión o detección de outliers [40]. Este algoritmo construye un hiperplano óptimo entre dos clases en forma de superficie de decisión, de modo que permite la predicción de nuevos ejemplos [41].

Para este modelo, se ajustan distintos parámetros para realizar el mejor entrenamiento posible. Por ello, se selecciona el parámetro 'kernel: rbf' y se dan los siguientes intervalos de valores para los parámetros 'gamma' y 'C':

'gamma':[1e-3,1e-2,0.1,0.5,1],'C':[0.1,0.5,1,10,100, 1000]

A continuación, se hace uso de la función GridsearchCV() que emplea el método cross-validation (CV) para determinar los parámetros que mejor se ajustan y conocer el mejor modelo de entrenamiento que se puede presentar empleando un valor de k = 10.

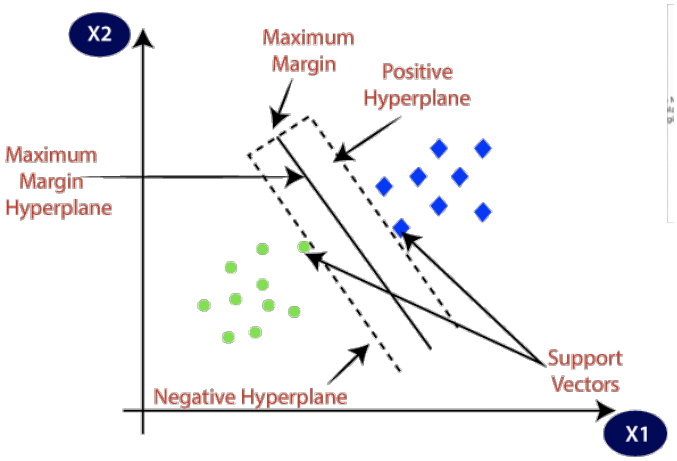


Fig. 8. Algoritmo Support Vector Machine [42].

2.4.2 Random Forest (RF)

El algoritmo Random Forest está formado por un conjunto de árboles de decisión individuales, cada uno entrenado con una muestra aleatoria extraída del dataset de entrenamiento [42]. Esto implica que cada árbol se entrena con unos datos ligeramente distintos. En cada árbol individual, las observaciones se van distribuyendo por nodos, generando la estructura del árbol hasta alcanzar un nodo terminal. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo [42].

En este caso, el parámetro que se ajusta es el número de árboles que se emplean: 'n_estimators':[10,25,50,100,200,500].

Del mismo modo, se emplea la función GridsearchCV() con el método 10fold-Cross-Validation para seleccionar el mejor parámetro que se ajusta a los datos de los modelos generados.

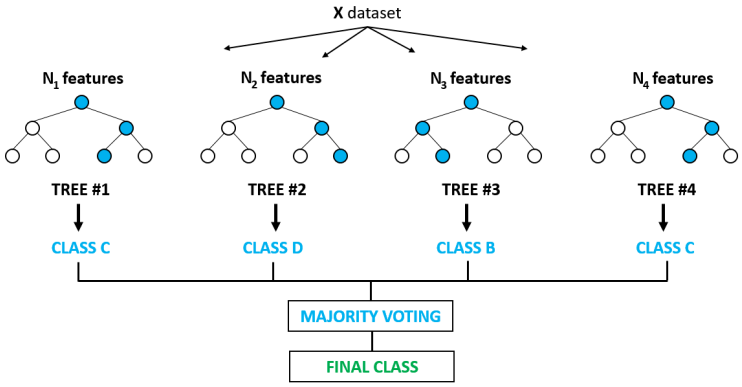


Fig. 9. Algoritmo Random Forest [43].

2.4.3 Gradient Boasted Tree (GBRT)

El algoritmo Gradient Boasted Tree está formado por un conjunto de árboles de decisión individuales, entrenados de forma secuencial, de forma que cada nuevo árbol trata de mejorar los errores de los árboles anteriores. La predicción de una nueva observación se obtiene agregando las predicciones de todos los árboles individuales que forman el modelo [44].

Para este caso, tenemos que ajustar los parámetros. El primero es el número de árboles (n_estimators) y el segundo es la tasa de aprendizaje (''learning_rate') que controla el grado en que cada árbol se le permite corregir los errores de los árboles anteriores, este parámetro lo configuraremos con un valor de 0.1. Para el parámetro 'n_estimators' seleccionamos el siguiente intervalo de valores:

{'n_estimators' : [50, 100, 200, 500]

Tal y como hemos comentado en los dos algoritmos anteriores, haciendo uso del método 10-fold-Cross-Validation con la función GridSearchCV(), seleccionamos los mejores parámetros que se ajustan para cada caso.

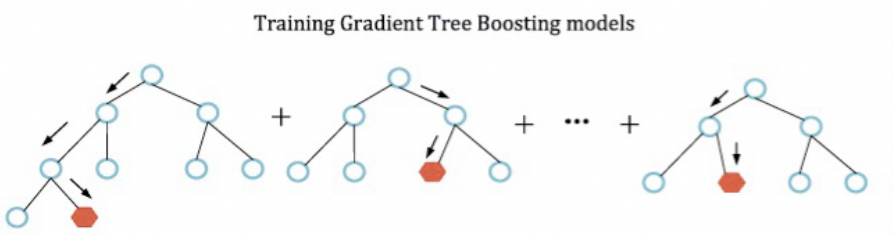


Fig. 10. Algoritmo Gradient Boasted Tree (GBRT) [45].

2.4.4 Evaluación de los modelos

Para la generación de cada modelo, primero se entrenan los modelos con el 80% de los datasets y luego, se testean con el 20% restante.

Tras realizar el testeo, evaluamos los modelos para ver cuál se ajusta mejor al estudio. Para ello, utilizamos la matriz de confusión y las métricas de calidad: exactitud, precisión, sensibilidad, especificidad, F1 y el área bajo la curva ROC (AUC).

· Matriz de confusión.

Es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna representa el número de predicciones de cada clase, mientras que cada fila representa las instancias en la clase real. Permite ver los tipos de aciertos y errores que tiene nuestro modelo cuando pasamos por el proceso de aprendizaje con los datos.

Fig. 11. Matriz de confusión [46].

A partir de las cuatro opciones que posee la matriz de confusión, se construyen los parámetros exactitud, precisión, sensibilidad, especificidad y F1 usados para evaluar la calidad del modelo.

- Exactitud (accuracy): mide el porcentaje de casos que el modelo ha acertado.

  Exactitud = (VP + VN) / (VP + VN + FP + FN)

- Precisión (precision): nos da la probabilidad de que, dada una predicción positiva, la realidad sea positiva también.

  Precisión = VP / (VP + FP)

- Sensibilidad (recall): es la proporción de casos positivos que fueron correctamente identificados por el algoritmo.

  Sensibilidad = VP / (VP + FN)

- Especificidad (especifity): se trata de los casos negativos que el algoritmo ha clasificado correctamente.

  Especificidad = VN / (VN + FP)

- F1 score: es otro parámetro muy usado porque nos resume la precisión y la sensibilidad en una sola métrica. Es de gran utilidad cuando la distribución de las clases es desigual.

  F1 = 2 * (sensibilidad * precisión) / (sensibilidad + precisión)

→ AUC de ROC.

Una curva ROC (curva de característica operativa del recepto) es un gráfico que muestra el rendimiento de un modelo de clasificación, y representa la tasa de verdaderos positivos frente a la tasa de falsos positivos. Por tanto, AUC mide todo el área bidimensional por debajo de la curva ROC completa [47]. El valor de este parámetro se puede interpretar como, dado dos péptidos, un tóxico y otro no tóxico, la prueba los clasifique correctamente. Cuanto mayor sea el valor de AUC, mejor será el ajuste del modelo generado. Este valor es también utilizado para evaluar la calidad del modelo predictivo.

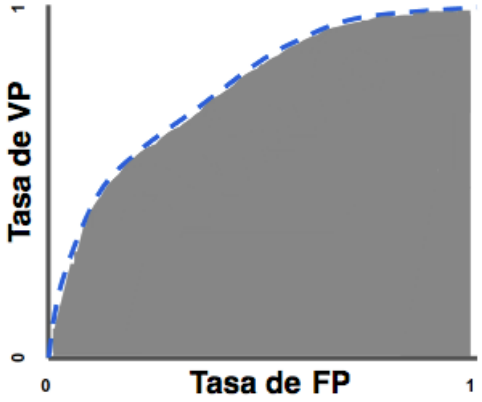


Fig. 12. Área bajo la curva (AUC) de ROC [47].

3. Resultados y Discusión

3.1. Parámetros de los algoritmos

En este capítulo vamos a mostrar los mejores parámetros obtenidos para cada modelo generado según el tipo algoritmo utilizado.

Tal y como se comentó en el apartado de Materiales y Métodos, se utiliza en cada algoritmo la función GridsearchCV() que emplea el método cross-validation (CV) para determinar los parámetros que mejor se ajustan, y conocer el mejor modelo de entrenamiento empleando un valor de k = 10.

3.1.1. Support Vector Machine (SVM)

Modelo	Kernel	C	Gamma
Dataset el mejor descriptor	rbf	1	1
Dataset balanceado Subsampling	rbf	10	0.01
Dataset DBSCAN	rbf	1	0.1
Dataset DBSCAN + Subsampling	rbf	100	0.01
Dataset Linclust/MMseqs2	rbf	1000	0.1

3.1.2. Random Forest (RF)

Modelo	n_estimators
Dataset inicial	10
Dataset el mejor descriptor	50
Dataset dos mejores descriptores	50
Dataset balanceado Subsampling	200
Dataset DBSCAN	10
Dataset DBSCAN + Subsampling	50
Dataset Linclust/MMseqs2	500

3.1.3. Gradient Boasted Tree (GBRT)

Modelo	n_estimators	learning_rate
Dataset inicial	10	0.1
Dataset el mejor descriptor	50	0.1
Dataset dos mejores descriptores	50	0.1
Dataset balanceado Subsampling	500	0.1
Dataset DBSCAN	50	0.1
Dataset DBSCAN + Subsampling	200	0.1
Dataset Linclust/MMseqs2	200	0.1

3.2. Evaluación de los modelos creados

Después de seleccionar los mejores parámetros para cada modelo, entrenamos cada uno con el 80% del dataset.

Tras ajustar el modelo, procedemos al testeo con el 20% del dataset restante que no se han empleado para entrenar cada uno de los modelos. En esta etapa, se evalúan los modelos mediante la matriz de confusión y las métricas de calidad que hemos explicado en el apartado de Materiales y Métodos.

Primero, vamos a realizar el algoritmo Support Vector Machine para cada dataset generado (explicados en el apartado '1.5 Breve sumario de productos obtenidos'). En este caso, como consecuencia de las grandes dimensiones del primer dataset (dataset con todos los descriptores PAAC de cada péptido) y del tercer dataset (compuesto por los dos mejores descriptores), se ha decidido no realizar el modelo SVM debido a problemas en tiempos de ejecución computacional.

· Algoritmo SVM:

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo mejor descriptor	0,98	1	0,98		0,99	0,5

Fig. 13. Resultados obtenidos de las métricas de calidad para el modelo creado con el algoritmo SVM a partir del dataset con el mejor descriptor PAAC.

Modelo		
Modelo mejor descriptor	NonToxic	Toxic
NonToxic	93856	0
Toxic	2276	0

Fig. 14. Matriz de confusión para el modelo creado con el algoritmo SVM a partir del dataset con el mejor descriptor PAAC.

Para los modelos generados con los algoritmos Random Forest y Gradient Boasted Tree no se han tenido problemas en tiempos de ejecución. Por ello, se representan todos los modelos predictivos a partir de los datasets creados.

· Algoritmo RF:

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo dataset inicial	0,98	1	0,99	0,77	0,99	0,75
Modelo mejor descriptor	0,97	1	0,98	0,11	0,99	0,51
Modelo 2 mejores descriptores	0,97	1	0,98	0,1	0,99	0,51

Fig. 15. Métricas de calidad obtenidas de los modelo RF.

Modelo		
Modelo dataset inicial	NonToxic	Toxic
NonToxic	93509	347
Toxic	1139	1137
Modelo mejor descriptor	NonToxic	Toxic
NonToxic	93509	347
Toxic	2233	43
Modelo 2 mejores descriptores	NonToxic	Toxic
NonToxic	93466	390
Toxic	2234	42

Fig. 16. Matriz de confusión de los modelos RF.

A

	precision	recall	f1-score	support
NonToxic	0.99	1.00	0.99	93856
Toxic	0.77	0.51	0.61	2276
accuracy			0.98	96132
macro avg	0.88	0.75	0.80	96132
weighted avg	0.98	0.98	0.98	96132

B

	precision	recall	f1-score	support
NonToxic	0.98	1.00	0.99	93856
Toxic	0.10	0.02	0.03	2276
accuracy			0.97	96132
macro avg	0.54	0.51	0.51	96132
weighted avg	0.96	0.97	0.96	96132

C

	precision	recall	f1-score	support
NonToxic	0.98	1.00	0.99	93856
Toxic	0.10	0.02	0.04	2276
accuracy			0.97	96132
macro avg	0.54	0.51	0.51	96132
weighted avg	0.96	0.97	0.96	96132

Fig. 17. Métricas de calidad por categoría (Toxic/NonToxic) de los modelos RF. A) Modelo con dataset inicial. B) Modelo con dataset con el mejor descriptor. C) Modelo con dataset con los dos mejores descriptores.

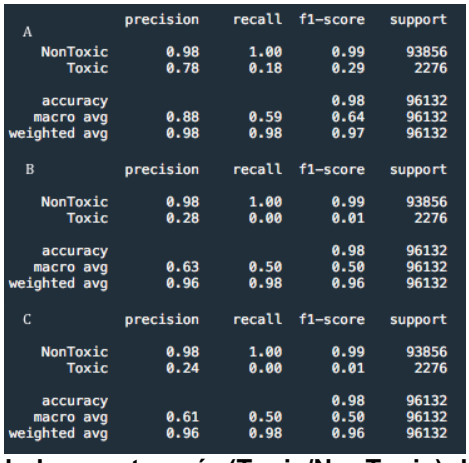
· Algoritmo GBRT:

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo dataset inicial	0,98	1	0,98	0,78	0,99	0,59
Modelo mejor descriptor	0,98	1	0,98	0,28	0,99	0,5
Modelo 2 mejores descriptores	0,98	1	0,98	0,24	0,99	0,5

Fig. 18. Métricas de calidad obtenidas de los modelo GBRT.

Modelo		
Modelo dataset inicial	NonToxic	Toxic
NonToxic	93742	114
Toxic	1877	399
Modelo mejor descriptor	NonToxic	Toxic
NonToxic	93835	21
Toxic	2268	8
Modelo 2 mejores descriptores	NonToxic	Toxic
NonToxic	93830	26
Toxic	2268	8

Fig. 19. Matriz de confusión de los modelos GBRT.



A

	precision	recall	f1-score	support
NonToxic	0.98	1.00	0.99	93856
Toxic	0.78	0.18	0.29	2276
accuracy			0.98	96132
macro avg	0.88	0.59	0.64	96132
weighted avg	0.98	0.98	0.97	96132

B

	precision	recall	f1-score	support
NonToxic	0.98	1.00	0.99	93856
Toxic	0.28	0.00	0.01	2276
accuracy			0.98	96132
macro avg	0.63	0.50	0.50	96132
weighted avg	0.96	0.98	0.96	96132

C

	precision	recall	f1-score	support
NonToxic	0.98	1.00	0.99	93856
Toxic	0.24	0.00	0.01	2276
accuracy			0.98	96132
macro avg	0.61	0.50	0.50	96132
weighted avg	0.96	0.98	0.96	96132

Fig. 20. Métricas de calidad por categoría (Toxic/NonToxic) de los modelos GBRT. A) Modelo con dataset inicial. B) Modelo con dataset con el mejor descriptor. C) Modelo con dataset con los dos mejores descriptores

Una vez realizados los tres primeros modelos, observamos que los resultados son de poca calidad. En general, los modelos generados clasifican muy bien los péptidos no tóxicos, sin embargo, no ocurre lo mismo con los péptidos tóxicos.

Observando la mayoría de los parámetros generales resultantes (Fig.13., Fig.15., Fig.18.), aparentemente los resultados son buenos y por tanto, los modelos realizan buenas predicciones. Sin embargo, esto puede llevar a una confusión. Con la ayuda de los resultados por categoría (Fig.17., Fig.20.), podemos ver que estos tipos de resultados no son fiables. Fijándonos en cualquier modelo que clasifique la categoría 'Toxic', podemos afirmar que son modelos que no están equilibrados a la hora de clasificar dos categorías, ya que predicen muy bien una, en este caso, la categoría 'NonToxic', pero la clasificación de la otra categoría es de poca calidad. Otra forma muy eficaz de observar la poca fiabilidad y calidad de los modelos es el parámetro AUC ya que da valores muy bajos.

Un claro ejemplo lo podemos observar cuando se realiza el modelo RF con el dataset inicial. Vemos como las métricas de calidad de la Fig. 15., aparentemente, dan unos resultados aceptables. Sin embargo, cuando analizamos los parámetros de evaluación para cada categoría (Fig. 18.), podemos ver que la categoría 'Toxic' presenta unos valores de recall y F1 bajos (0.51 y 0.61, respectivamente). Este caso es un ejemplo típico de modelos con desequilibrio, donde se obtiene un alto valor de precisión en la clase Mayoritaria y un bajo recall en la clase Minoritaria [27].

En nuestro caso, el dataset creado presenta una gran cantidad de péptidos no tóxicos y muy pocos péptidos tóxicos, por lo que nos encontramos ante el problema de un dataset en desequilibrio.

Por ello, se ha optado por realizar distintas técnicas para resolver el problema de desequilibrio del dataset y volver a generar modelos predictivos para ver si los resultados mejoran.

Las técnicas realizadas y ya comentadas son:

3.2.1. Subsampling de la clase mayoritaria

En este caso, se parte del conjunto de datos de péptidos con todos los descriptores PAAC, es decir, 480656 péptidos (11364 tóxicos y 469292 no tóxicos).

Mediante la función NearMiss() de la librería Imblearn de Python se reduce el número de observaciones de la clase mayoritaria hasta obtener el mismo número de observaciones de la clase minoritaria. En nuestro caso, se reduce la clase de péptidos no tóxicos hasta obtener 11364 péptidos. De esta forma, resolvemos el problema de desequilibrio y podemos generar modelos predictivos con el dataset de 22728 péptidos (11364 péptidos tóxicos y 11364 péptidos no tóxicos).

3.2.2. Clustering DBSCAN

Para el agrupamiento de péptidos a partir de las características cuantitativas de los pseudo-aminoácidos y la posterior búsqueda de outliers, iFeature presenta la función DBSCAN, que dado un fichero con los descriptores PAAC de cada péptido en formato fasta, agrupa los péptidos en distintos clústeres según los valores de los descriptores PAAC. La función se puede encontrar en GitHub con el nombre 'funcion_DBSCAN.py'.

Tras ejecutar la función y eliminar los 446726 outliers resultantes, obtenemos un dataset de 33930 péptidos, 33227 no tóxicos y 703 péptidos tóxicos, con el que vamos a generar modelos predictivos con los 3 algoritmos Machine Learning explicados.

Observamos que el dataset sigue estando desbalanceado (clase mayoritaria: péptidos no tóxicos) lo que puede provocar unos resultados de predicción similares a los obtenidos hasta el momento. En este caso, optamos por realizar un balanceo del dataset resultante mediante la primera técnica comentada (Subsampling de la clase mayoritaria). De esta forma, obtenemos un nuevo dataset que se compone de 703 péptidos de cada clase. Este dataset también lo usaremos para generar modelos predictivos.

3.2.3. Clustering Linclust/MMseqs2

Tal y como se ha comentado en el apartado Materiales y Métodos, se ha realizado un clustering por secuencia a partir del dataset inicial de 480656 péptidos. Se ha utilizado el software MMseqs2 con la opción 'easy-linclust', una cobertura de 0.6 y una identidad del 100%, de esta forma, nos aseguramos que todos los péptidos que conforme un clúster son idénticos entre sí.

Hemos hecho uso de este software con la ayuda del terminal, escribiendo la siguiente línea de comandos:

```
mmseqs easy-linclust examples/All_peptides.fasta clusterRes tmp --min-seq-id 1.0 -c 0.6 --cov-mode 1
```

Fig. 21. Comando utilizado para realizar el clustering por secuencias Linclust/MMseqs2.

Finalmente, obtenemos un fichero que muestra dos columnas. La primera columna muestra el clúster representativo y la segunda muestra el clúster miembro. Contamos el número de péptidos que no forman clúster con ningún otro péptido, es decir, identificamos los 'single' péptidos y los consideraremos outliers.

Tras eliminar los outliers, obtenemos un dataset de 7666 péptidos no tóxicos y 8067 tóxicos que también será usado para generar los modelos.

Resumidamente, hemos creado los cuatro siguientes datasets que serán utilizados para generar nuevos modelos predictivos:

· Dataset balanceado a partir del dataset inicial (480656 péptidos) con la estrategia Subsampling.

Detalles:

- Nombre de los modelos que realizamos con este dataset: **'Modelo Subsampling'**.
- Composición: **22728 péptidos** (11364 péptidos tóxicos y 11364 péptidos no tóxicos).

· Dataset resultante del clustering DBSCAN a partir del dataset inicial (480656 péptidos).

Detalles:

- Nombre de los modelos que realizamos con este dataset: **'Modelo DBSCAN'**.
- Composición: **33930 péptidos** (33227 péptidos tóxicos y 703 péptidos no tóxicos).

· Dataset en el que primero se realizó un clustering DBSCAN a partir del dataset inicial (480656 péptidos) y posteriormente se balanceó mediante la estrategia Subsampling.

Detalles:

- Nombre de los modelos que realizamos con este dataset: **'Modelo DBSCAN+Subsampling'**.
- Composición: **1406 péptidos** (703 péptidos tóxicos y 703 péptidos no tóxicos).

· Dataset resultante del clustering Linclust por secuencias peptídicas.

Detalles:

- Nombre de los modelos que realizamos con este dataset: **'Modelo Linclust**.
- Composición: **15733 péptidos** (8067 péptidos tóxicos y 7666 péptidos no tóxicos).

Una vez entrenados los modelos predictivos para cada dataset, se testean para evaluarlos y observar su calidad de predicción.

3.2.4. Resultados obtenidos

3.2.4.1. Resultados con el algoritmo SVM

A continuación, vamos a mostrar los resultados que hemos obtenido para cada modelo generado con el algoritmo SVM.

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo Subsampling	0,86	0,9	0,83	0,89	0,86	0,86
Modelo DBSCAN	0,99	1	0,99	1	1	0,8
Modelo DBSCAN+SUBSAMPLING	0,9	0,93	0,87	0,94	0,9	0,91
Modelo Linclust	0,96	0,95	0,97	0,95	0,96	0,96

Fig. 22. Resultados obtenidos de las métricas de calidad para los modelos creados con el algoritmo SVM.

Modelo		
Modelo Subsampling	NonToxic	Toxic
NonToxic	2002	227
Toxic	420	1897
Modelo DBSCAN	NonToxic	Toxic
NonToxic	6645	0
Toxic	57	84
Modelo DBSCAN+SUBSAMPLING	NonToxic	Toxic
NonToxic	125	9
Toxic	18	130
Modelo Linclust	NonToxic	Toxic
NonToxic	1457	75
Toxic	38	1577

Fig. 23. Matriz de confusión de los modelos creados con el algoritmo SVM.

A

	precision	recall	f1-score	support
NonToxic	0.83	0.90	0.86	2229
Toxic	0.89	0.82	0.85	2317
accuracy			0.86	4546
macro avg	0.86	0.86	0.86	4546
weighted avg	0.86	0.86	0.86	4546

B

	precision	recall	f1-score	support
NonToxic	0.99	1.00	1.00	6645
Toxic	1.00	0.60	0.75	141
accuracy			0.99	6786
macro avg	1.00	0.80	0.87	6786
weighted avg	0.99	0.99	0.99	6786

C

	precision	recall	f1-score	support
NonToxic	0.87	0.93	0.90	134
Toxic	0.94	0.88	0.91	148
accuracy			0.90	282
macro avg	0.90	0.91	0.90	282
weighted avg	0.91	0.90	0.90	282

D

	precision	recall	f1-score	support
NonToxic	0.97	0.95	0.96	1532
Toxic	0.95	0.98	0.97	1615
accuracy			0.96	3147
macro avg	0.96	0.96	0.96	3147
weighted avg	0.96	0.96	0.96	3147

Fig. 24. Resultados por categoría de los modelos generados con el algoritmo SVM. A) Modelo Subsampling SVM. B) Modelo DBSCAN SVM. C) Modelo DBSCAN+Subsampling SVM. D) Modelo Linclust SVM.

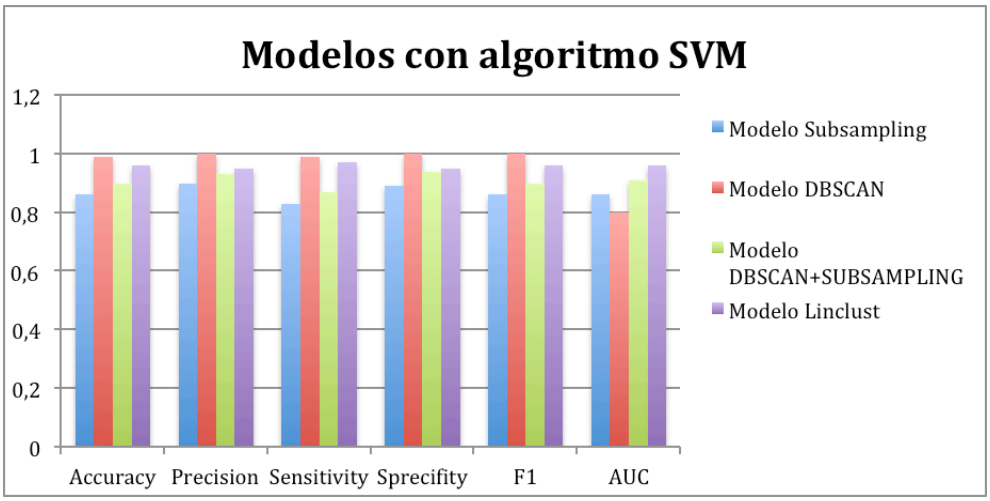


Fig. 25. Diagrama de barras con valores de parámetros de calidad para cada modelo. Algoritmo SVM.

Podemos apreciar que el modelo 'Modelo Linclust' es el que mejor se ajusta cuando usamos el algoritmo SVM, seguido del 'Modelo DBSCAN+Subsampling. También podemos apreciar en el diagrama de barras que el 'Modelo DBSCAN' presenta buenos resultados, pero tal y como ha ocurrido en los anteriores casos, para la clase 'Toxic' el modelo no clasifica tan bien como los 3 restantes (Fig. 24. B). Esto puede ser debido a la composición del dataset, ya que tras realizar la función DBSCAN y eliminar los outliers, nos queda un dataset con una diferencia muy amplia entre el número de péptidos tóxicos y no tóxicos.

3.2.4.2. Resultados con el algoritmo RF

Mostramos los resultados que se han obtenido para los modelos generados con el algoritmo RF.

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo Subsampling	0,86	0,93	0,81	0,92	0,87	0,86
Modelo DBSCAN	0,99	1	0,99	0,72	0,99	0,77
Modelo DBSCAN+Subsampling	0,88	0,97	0,81	0,97	0,88	0,88
Modelo Linclust	0,98	0,98	0,97	0,98	0,98	0,98

Fig. 26. Resultados obtenidos para los modelos creados con el algoritmo RF.

Modelo		
Modelo Subsampling	NonToxic	Toxic
NonToxic	2066	163
Toxic	477	1840
Modelo DBSCAN	NonToxic	Toxic
NonToxic	6615	30
Toxic	63	78
Modelo DBSCAN+Subsampling	NonToxic	Toxic
NonToxic	130	4
Toxic	30	118
Modelo Linclust	NonToxic	Toxic
NonToxic	1496	36
Toxic	40	1575

Fig. 27. Matriz de confusión de los modelos creados con el algoritmo RF.

A

	precision	recall	f1-score	support
NonToxic	0.82	0.93	0.87	2229
Toxic	0.92	0.80	0.86	2317
accuracy			0.86	4546
macro avg	0.87	0.87	0.86	4546
weighted avg	0.87	0.86	0.86	4546

B

	precision	recall	f1-score	support
NonToxic	0.99	1.00	0.99	6645
Toxic	0.72	0.55	0.63	141
accuracy			0.99	6786
macro avg	0.86	0.77	0.81	6786
weighted avg	0.98	0.99	0.99	6786

C

	precision	recall	f1-score	support
NonToxic	0.81	0.97	0.88	134
Toxic	0.97	0.80	0.87	148
accuracy			0.88	282
macro avg	0.89	0.88	0.88	282
weighted avg	0.89	0.88	0.88	282

D

	precision	recall	f1-score	support
NonToxic	0.97	0.98	0.98	1532
Toxic	0.98	0.98	0.98	1615
accuracy			0.98	3147
macro avg	0.98	0.98	0.98	3147
weighted avg	0.98	0.98	0.98	3147

Fig. 28. Resultados por categoría de los modelos generados con el algoritmo RF. A) Modelo Subsampling RF. B) Modelo DBSCAN RF. C) Modelo DBSCAN+Subsampling RF. D) Modelo Linclust RF.

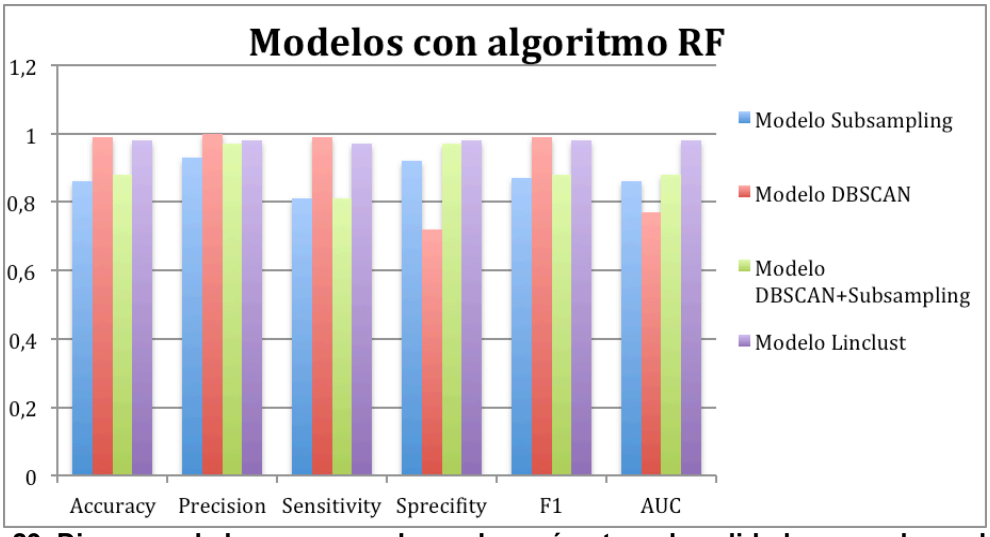


Fig. 29. Diagrama de barras con valores de parámetros de calidad para cada modelo. Algoritmo RF.

Para este caso, obtenemos resultados similares a los que se obtuvieron con el algoritmo SVM. Tenemos el modelo 'Modelo Linclust' que presenta los mejores indicadores de calidad, seguido del modelo 'DBSCAN+Subsampling'. Además, observamos el mismo problema de desequilibrio del dataset en el modelo 'Modelo DBSCAN', tal y como ocurre con el algoritmo SVM.

3.2.4.3. Resultados con el algoritmo GBRT

Mostramos los resultados que se han obtenido para los modelos generados con el algoritmo RF.

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo Subsampling	0,84	0,89	0,81	0,89	0,85	0,84
Modelo DBSCAN	0,99	1	0,99	0,95	0,99	0,75
Modelo DBSCAN+Subsampling	0,9	0,92	0,88	0,92	0,9	0,9
Modelo Linclust	0,91	0,9	0,9	0,91	0,9	0,91

Fig. 30. Resultados obtenidos para los modelos creados con el algoritmo GBRT.

Modelo		
Modelo Subsampling	NonToxic	Toxic
NonToxic	1989	240
Toxic	469	1848
Modelo DBSCAN	NonToxic	Toxic
NonToxic	6641	4
Toxic	71	70
Modelo DBSCAN+Subsampling	NonToxic	Toxic
NonToxic	123	11
Toxic	16	132
Modelo Linclust	NonToxic	Toxic
NonToxic	1385	147
Toxic	150	1465

Fig. 31. Matriz de confusión de los modelos creados con el algoritmo GBRT.

A

	precision	recall	f1-score	support
NonToxic	0.81	0.89	0.85	2229
Toxic	0.89	0.80	0.84	2317
accuracy			0.84	4546
macro avg	0.85	0.84	0.84	4546
weighted avg	0.85	0.84	0.84	4546

B

	precision	recall	f1-score	support
NonToxic	0.99	1.00	0.99	6645
Toxic	0.95	0.50	0.65	141
accuracy			0.99	6786
macro avg	0.97	0.75	0.82	6786
weighted avg	0.99	0.99	0.99	6786

C

	precision	recall	f1-score	support
NonToxic	0.88	0.92	0.90	134
Toxic	0.92	0.89	0.91	148
accuracy			0.90	282
macro avg	0.90	0.90	0.90	282
weighted avg	0.90	0.90	0.90	282

D

	precision	recall	f1-score	support
NonToxic	0.90	0.90	0.90	1532
Toxic	0.91	0.91	0.91	1615
accuracy			0.91	3147
macro avg	0.91	0.91	0.91	3147
weighted avg	0.91	0.91	0.91	3147

Fig. 32. Resultados por categoría de los modelos generados con el algoritmo GBRT. A) Modelo Subsampling GBRT. B) Modelo DBSCAN GBRT. C) Modelo DBSCAN+Subsampling SVM. D) Modelo Linclust GBRT.

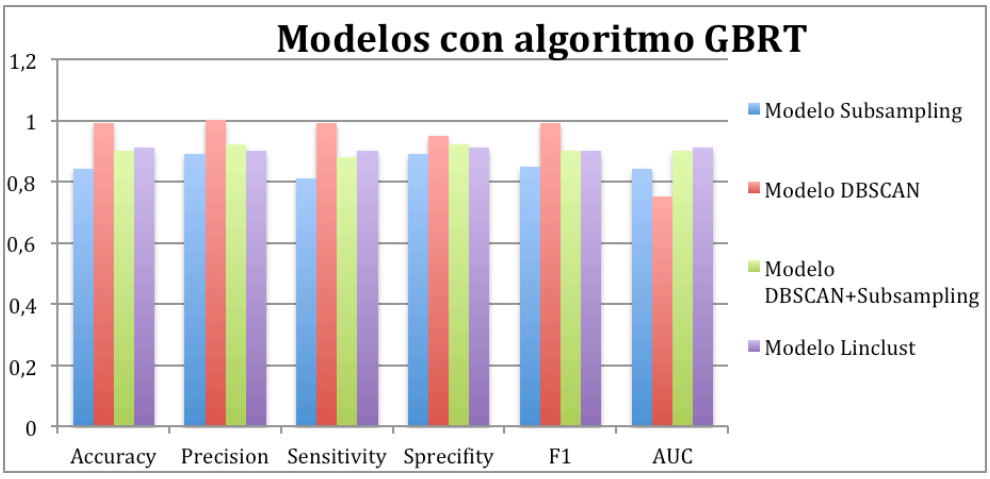


Fig. 33. Diagrama de barras. Métricas de calidad para cada modelo. Algoritmo RF.

Para este último algoritmo podemos ver, a partir de la Fig.33., que los modelos Linclust y DBSCAN+Subsampling presentan resultados muy similares.

Tras evaluar los modelos realizados con los 3 algoritmos, buscamos modelos con un equilibrio entre los parámetros de evaluación para la clasificación peptídica, es decir, queremos modelos que clasifiquen de forma correcta tanto los péptidos tóxicos como los no tóxicos.

Tras la etapa del testeo, observamos que los modelos 'Modelo Linclust' para cada algoritmo ML realizan las mejores predicciones de toxicidad según los indicadores de calidad. Además, observamos que dependiendo del tipo de dataset utilizado, un algoritmo se ajusta mejor que otro. Por tanto, para el modelo Subsampling los algoritmos SVM y RF se ajustan mejor. Mientras que para el dataset 'DBSCAN+Subsampling', podemos seleccionar los algoritmos SVM o GBRT, ya que los modelos realizados presentan resultados muy similares a la hora de hacer predicciones. El algoritmo RF es el que mejor se ajusta al dataset Linclust ('Modelo Linclust').

Para cada algoritmo, podemos ver cómo hay modelos que clasifican mejor los péptidos tóxicos y otros, en cambio, los péptidos no tóxicos. Esto puede ser debido a la composición peptídica de cada dataset utilizado para generar los modelos predictivos. Por ejemplo, en el caso del modelo 'Modelo Linclust', el dataset utilizado tiene un mayor número de péptidos con la categoría de tóxico, por ello es el modelo que, según los resultados explicados hasta el momento, mejor clasifica a los péptidos tóxicos.

Por último, concluimos que descartamos los modelos 'Modelos DBSCAN' generados con los tres algoritmos, ya que buscamos modelos que tengan un equilibrio a la hora de clasificar tanto a los péptidos tóxicos como a los no tóxicos. Observando la matriz de confusión de cada modelo, vemos que se predicen muchos péptidos tóxicos como no tóxicos, de ahí, que los valores de 'recall' y 'F1' en los resultados por categoría (Fig. 24., Fig. 28. y Fig.32.) sean bajos. Concluimos que estos modelos creados a partir del clustering DBSCAN del dataset inicial no son fiables ya que los datasets utilizados para generar los modelos presentan un desequilibrio de clases.

3.3. Evaluación de los modelos predictivos a partir de un dataset externo

En este apartado se han seleccionado los mejores modelos predictivos generados y se ha creado un dataset externo con péptidos que no están presentes en el dataset inicial de 480656 péptidos.

Los modelos seleccionados son los siguientes:

- Modelo Subsampling con el algoritmo SVM.
- Modelo Subsampling con el algoritmo RF.
- Modelo DBSCAN+Subsampling con el algoritmo SVM.
- Modelo DBSCAN+Subsampling con el algoritmo GBRT.
- Modelo Linclust con el algoritmo RF.

Seleccionamos modelos con el mismo dataset pero con distinto algoritmo Machine Learning porque presentan resultados muy similares pero hay modelos que clasifican mejor la categoría 'Toxic' de los péptidos tóxicos y otros, la categoría 'NonToxic'.

Queremos probar, dado un conjunto de péptidos nuevo, si los modelos seleccionados clasifican de forma adecuada los péptidos.

El dataset está compuesto de 11 péptidos, 5 tóxicos y 6 no tóxicos, recopilados de la base de datos PlantPepDB [48]. Hemos hecho uso de la función PAAC de iFeature para obtener sus características cuantitativas y hemos utilizado los mejores modelos predictivos para predecir estos nuevos péptidos y clasificarlos como tóxicos o no tóxicos.

De la misma forma que en el apartado anterior, evaluamos los modelos mediante la matriz de confusión y las métricas de calidad: exactitud, precisión, sensibilidad, especificidad, F1 y AUC de la curva ROC.

Modelo	Accuracy	Precision	Sensitivity	Sprecifity	F1	AUC
Modelo Subsampling SVM	1	1	1	1	1	1
Modelo Subsampling RF	1	1	1	1	1	1
Modelo DBSCAN+SUBSAMPLING SVM	0,64	0,33	1	0,56	0,5	0,67
Modelo DBSCAN+Subsampling GBRT	1	1	1	1	1	1
Modelo Linclust RF	0,36	0	0	0,4	0	0,4

Fig. 34. Parámetros para evaluar la calidad de predicción de los mejores modelos predictivos.

Modelo		
Modelo Subsampling SVM	NonToxic	Toxic
NonToxic	6	0
Toxic	0	5
Modelo Subsampling RF	NonToxic	Toxic
NonToxic	6	0
Toxic	0	5
Modelo DBSCAN+SUBSAMPLING SVM	NonToxic	Toxic
NonToxic	2	4
Toxic	0	5
Modelo DBSCAN+Subsampling GBRT	NonToxic	Toxic
NonToxic	6	0
Toxic	0	5
Modelo Linclust RF	NonToxic	Toxic
NonToxic	0	6
Toxic	1	4

Fig. 35. Matriz de confusión de los mejores modelos predictivos.

Basándonos en los resultados obtenidos, observamos que los mejores modelos son: 'Modelo Subsampling SVM', 'Modelo Subsampling RF' y 'Modelo DBSCAN+Subsampling GBRT', ya que además de obtener buenos resultados en la etapa de testeo cuando generamos el modelo, también clasifican correctamente todos los péptidos del dataset externo.

Por otro lado, aunque el modelo 'Modelo Linclust RF' realiza una buena predicción en la etapa del testeo con el 20% del dataset, cuando se utiliza un dataset externo que contiene péptidos diferentes no realiza una buena clasificación, tal y como podemos ver en la Fig. 35. (clasifica mal 6 péptidos no tóxicos). Esto puede ser debido a la técnica de clustering Linclust que realizamos para crear el dataset con el que se generó el modelo. Ya que, después de diversas pruebas, la única forma de obtener un dataset equilibrado es configurando el parámetro de identidad al 100%. De esta manera, todos los clústeres que se realizan tienen un 100% de identidad entre los péptidos. Esta alta selectividad podría ser el problema de la mala calidad de predicción del modelo.

Por último, podemos ver que el modelo 'Modelo DBSCAN+Subsampling SVM' predice de forma correcta los péptidos tóxicos pero no ocurre lo mismo con los péptidos no tóxicos. La razón de estos resultados podría deberse a la composición del dataset utilizado y el algoritmo utilizado. El algoritmo SVM no se ajustaría adecuadamente al dataset en estudio. Observamos que para el mismo dataset pero utilizando un algoritmo diferente (algoritmo GBRT) los resultados de predicción son mejores, posiblemente porque el modelo creado al utilizar el algoritmo GBRT se ajusta mejor a los datos.

Tras los distintos resultados obtenidos, podemos concluir que una vez entrenado el modelo predictivo con el 80% de los datos y testeado con el 20% restante, es necesario extrapolarlo a otros datasets externos para asegurar una correcta clasificación del modelo y dar una fiabilidad de que el modelo se puede aplicar para distintos péptidos.

4. Conclusiones

A continuación, tras haber analizado los resultados, vamos a comentar las conclusiones a las que hemos llegado:

1) El tratamiento del dataset de péptidos con los descriptores PAAC (dataset de 480656 péptidos) mediante la técnica de balanceo Subsampling o las técnicas de clustering DBSCAN o Linclust generan modelos predictivos de mayor calidad en comparación con los modelos generados con el dataset inicial o con los datasets seleccionando los mejores descriptores mediante la estrategia chi-cuadrado.
2) El problema de desequilibrio del dataset inicial se resuelve con la técnica de balanceo Subsampling o las técnicas de clustering DBSCAN o Linclust.
3) Cada algoritmo de ML se ajusta mejor a un dataset u otro dependiendo de si ha sido tratado con la técnica Subsampling o técnicas de clustering:

    a. Para el dataset balanceado mediante la estrategia Subsampling, los algoritmos SVM o RF obtienen mejores modelos de predicción.
    b. Los algoritmos SVM y GBRT se ajustan mejor a los datos resultantes de tratar el dataset inicial mediante el clustering DBSCAN y el posterior balanceo con la estrategia Subsampling.
    c. El dataset obtenido del clustering por secuencia (clustering Linclust) obtiene resultados de mayor calidad cuando se utiliza el algoritmo RF.

4) Los modelos ‘Modelo Subsampling SVM’, ‘Modelo Subsampling RF’ y ‘Modelo DBSCAN+Subsampling GBRT’ son los mejores modelos generados, ya que obtienen resultados de calidad cuando predicen los resultados de la parte test (20% del dataset) y clasifican correctamente los péptidos del dataset externo.
5) Entre las técnicas utilizadas para tratar el dataset, concluimos que la estrategia Subsampling podría ser la alternativa más fiable para tratar el dataset desequilibrado y obtener modelos de mayor calidad.
6) Necesidad de valorar los modelos con datasets externos para asegurar la fiabilidad de clasificación de toxicidad, tal y como ocurre con el modelo Linclust utilizando el algoritmo RF.

4.1. Análisis de la planificación y metodología

En el presente trabajo se introdujo un cambio una vez realizados los primeros modelos predictivos, ya que los datasets iniciales estaban en desequilibrio. Para solventarlo, se plantearon distintas técnicas de balanceo y de clustering para tratar el dataset y crear datasets equilibrados y sin outliers. A partir de estos datasets se generaron nuevos modelos que tras obtener los resultados, se vio que realizaban buenas predicciones, por lo que se resolvió el problema de desequilibrio del dataset.

A pesar del cambio comentado, la planificación y metodología del proyecto han sido adecuadas. En un primer momento, se le dedicó más tiempo a la realización de las tareas de creación de los distintos datasets para posteriormente crear nuevos modelos predictivos. Sin embargo, para otras tareas como el entrenamiento de los modelos predictivos no se dedicó menos tiempo del establecido en el Diagrama de Gantt del apartado '1.4 Planificación del trabajo'. Por este motivo, la planificación temporal del proyecto se vio equilibrada y no fue necesario realizar una actualización del cronograma.

4.2. Trabajo futuro

La línea a seguir en este trabajo se basa en la necesidad de poder extrapolar estos modelos predictivos a todos los conjuntos de péptidos que existen, es decir, poder concluir con toda fiabilidad que la clasificación que realizan es correcta y aplicable a cualquier conjunto.

Esto, junto con la pérdida de globalidad (reducción del número de péptidos del dataset inicial para obtener modelos de mejor calidad), y el hecho de que para un mismo dataset cada algoritmo de Machine Learning clasifica mejor una categoría que otra, nos lleva a pensar en la utilización del método 'Ensamble learning' [49] o 'MulTask Learning (MTL)' [50].

Los métodos 'Ensamble learning' ayudan a mejorar los resultados del aprendizaje automático mediante la combinación de varios modelos [49, 51]. Utilizan múltiples algoritmos de aprendizaje para obtener un mejor rendimiento predictivo en comparación con el que se obtendría con un solo modelo predictivo [51].

MultiTaks Learning es un procedimiento para entrenar una red neuronal para que aprenda varias tareas relacionadas de forma simultánea considerando una de ellas como tarea principal y las otras como tareas secundarias [52].

5. Glosario

- PAAC: Descriptores pseudo-aminoácidos.
- SVM: Support Vector Machine.
- RF: Random Forest.
- GBRT: Gradient Boasted Tree.
- FN: Falso Negativo.
- FP: Falso Positivo.
- VP: Verdadero Positivo.
- VN: Verdadero Negativo.
- RBF: kernel de función de base radial.
- AUC: Área bajo la curva.
- ROC: Receiver Operating Characteristic.

6. Bibliografía

[1] Schmid, E.F. and Smith, D.A. (2005) Is declining innovation in the pharmaceutical industry a myth? Drug Discov. Today 10, 1031–1039.

[2] Kola, I. and Landis, J. (2004) Can the pharmaceutical industry reduce attrition rates? Nat. Rev. Drug Discov. 3, 711–715.

[3] Pangalos, M.N. et al. (2007) Drug development for CNS disorders: strategies for balancing risk and reducing attrition. Nat. Rev. Drug Discov. 6, 521–532.

[4] Rawlins, M.D. (2004) Cutting the cost of drug development? Nat. Rev. Drug Discov. 3, 360–364.

[5] Patrick Vlieghe, Vincent Lisowski, Jean Martinez and Michel Khrestchatisky (2010) Synthetic therapeutic peptides: science and market. ELSEVIER. Rev. Vol. 15.

[6] Vlieghe P, Lisowski V, Martinez J, Khrestchatisky M (2010) Synthetic therapeutic peptides: science and market. Drug Discov Today 15: 40–56.
[7] Thundimadathil J (2012) Cancer treatment using peptides: current therapies and future prospects. J Amino Acids 2012: 967347.

[8] Wu, Qihui; Ke, Hanzhong; Li, Dongli; Wang, Qi; Fang, Jiansong; Zhou, Jingwei (2019) Recent Progress in Machine Learning-based Prediction of Peptide Activity for Drug Discovery. Current Topics in Medicinal Chemistry, Volume 19, Number 1, pp. 4-16 (13).

[9] Chen F, Ma B, Yang ZC, Lin G, Yang D (2012) Extraordinary metabolic stability of peptides containing alpha-aminoxy acids. Amino Acids 43: 499–503.

[10] In Silico Approach for Predicting Toxicity of Peptides and Proteins, Gupta, S.; Kapoor, P.; Chaudhary, K.; Gautam, A.; Kumar, R.; Raghava, G. P. S. *PLoS One* 2013, *8*.

[11] “Taller avanzado para bioinformáticos” [online] Available: http://bioinformatica.uab.cat/base/base3.asp?sitio=sgbcursos&anar=otros&item=python [Accessed: 23/Dec/2020].

[12] “¿Es Python el lenguaje del futuro?” [online] Available: https://www.paradigmadigital.com/dev/es-python-el-lenguaje-del-futuro/ [Accesed: 9/Dec/2020].

[13] Zhen Chen, et al. iFeature: a python package and web server for features extraction and selection from protein and peptide sequences. Bioinformatics. 2018 Jul 15;34(14):2499-2502. doi: 10.1093/bioinformatics/bty140. PMID: 29528364; PMCID: PMC6658705.

[14] “ToxinPred Designing and prediction of toxic peptides” [online] Available: https://webs.iiitd.edu.in/raghava/toxinpred/dataset.php [Accessed: 27/Oct/2020].

[15] “Arachnoserver Spider toxin database” [online] Available: http://www.arachnoserver.org/downloadsequences.html [Accessed: 27/Oct/2020].

[16] “UniProtKB” [online] Available: https://www.uniprot.org/uniprot/?query=keyword:%22Toxin%20[KW-0800]%22&fil=reviewed%3Ayes&sort=score [Accessed: 27/Oct/2020].

[17] “UniProtKB” [online] Available: https://www.uniprot.org/uniprot/?query=NOT%20keyword:%22Toxin%20[KW-0800]%22&fil=reviewed%3Ayes&sort=score [Accessed: 27/Oct/2020].

[18] “Celiac Database” Version 2: peptides 1013: 18 January, 2018. [online] Available: http://www.allergenonline.org/celiacbrowse.shtml [Accessed: 27/Oct/2020].

[19] “NTXPred dataset” [online] Available: https://webs.iiitd.edu.in/raghava/ntxpred/dataset.html [Accessed: 27/Oct/2020].

[20] “BTXpred Supplementary Page” [online] Available: https://webs.iiitd.edu.in/raghava/btxpred/supplementary.html [Accessed: 27/Oct/2020].

[21] “KALIUM. Database of polypeptide ligands of potassium channels” [online] Available: https://kaliumdb.org [Accessed: 27/Oct/2020]

[22] “Toxin and Toxin Target Database (T3DB)” [online] Available: http://www.t3db.ca/downloads [Accessed: 27/Oct/2020].

[23] “DBETH, Database of Bacterial ExoToxins for Human” [online] Available: http://www.hpppi.iicb.res.in/btox/ [Accessed: 27/Oct/2020].

[24] Chou KC (May 2001). “Prediction of protein cellular attributes using pseudo-amino acid composition”. Proteins. 43 (3): 246-55. Doi:10.1002/prot.1035. PMID 11288174.

[25] “Type 1 pseudo amino acid composition” [online] Available: http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm [Accessed: 1/Dec/2020].

[26] “Prueba de chi-cuadrado (χ^2)_ qué es y cómo se usa en estadística” [online] Available: https://psicologiaymente.com/miscelanea/prueba-chi-cuadrado [Accessed: 2/Dec/2020].

[27] “Aprende Machine Learning, clasificación con datos desbalanceados” [online] Available: https://www.aprendemachinelearning.com/clasificacion-con-datos desbalanceados/ [Accessed: 11/Dec/2020].

[28] “Ejemplo de uso de DBSCAN en Python para eliminación de outliers” [online] Available: http://exponentis.es/ejemplo-de-uso-de-dbscan-en-python-para-deteccion-de-outliers [Accessed: 11/Dec/2020].

[29] “MMseqs2: ultra fast and sensitive sequence search and clustering suite” [online] Available: https://github.com/soedinglab/MMseqs2 [Accessed: 16/Dec/2020].

[30] Steinegger, M., Söding, J. Clustering huge protein sequence sets in linear time. *Nat Commun* **9,** 2542 (2018). https://doi.org/10.1038/s41467-018-04964-5

[31] Sheetlin, S., Park, Y., Frith, M. C. & Spouge, J. L. ALP & FALP: C++ libraries for pairwise local alignment E-values. *Bioinformatics* **32**, 304–305 (2015).

[32] Martin Steinegger, Milot Mirdita, Eli Levy Karin, Lars von den Driesch, Clovis Galiez, Johannes Söding. MMseqs2 User Guide.
[33] “Python” [online] Available: https://www.python.org [Accessed: 15/Oct/2020].

[34] “What is Python? Executive Summary [online] Available: https://www.python.org/doc/essays/blurb/ [Accessed: 15/Oct/2020].

[35] “SPYDER, The Scientific Python Development Environment” [online] Available: https://www.spyder-ide.org [Accessed: 16/Oct/2020].

[36] “iFeature Wev Server” [online] Available: https://ifeature.erc.monash.edu [Accessed: 16/Oct/2020].

[37] “MMseqs2, bio.tools” [online] Available: https://bio.tools/MMseqs2 [Accessed: 21/Oct/2020].

[38] “Hello world · GitHub Guides” [online] Available: https://guides.github.com/activities/hello-world/ [Accessed: 2/Jan/2021].

[39] ”GitHub” [online] Available: https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores [Accessed: 2/Jan/2021].

[40] “Support Vector Machines” [online] Available: https://scikit-learn.org/stable/modules/svm.html [Accessed: 25/Oct/2020].

[41] “Support Vector Machines Algorithm” [online] Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm [Accessed: 26/Oct/2020].

[42] Árboles de decisión, random forest, gradient boosting y C5.0. Joaquín Amat Rodrigo. Febrero, 2017. [online] Available: https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50 [Accessed: 29/Oct/2020].

[43] “Técnica Random Forest” [online] Available: https://rpubs.com/Avalos42/randomforest [Accessed: 29/Oct/2020].

[44] “Gradient Boosting con Python” [online] Available: https://www.cienciadedatos.net/documentos/py09_gradient_boosting_python.html [Accessed: 5/Dec/2020].

[45] Liu H, Zhang W, Nie L, Ding X, Luo J, Zou L. Predicting effective drug combinations using gradient tree boosting based on features extracted from drug-protein heterogeneous network. *BMC Bioinformatics*. 2019;20(1):645. Published 2019 Dec 9. doi:10.1186/s12859-019-3288-1.

[46] “La matriz de confusión y sus métricas” [online] Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ [Accessed: 5/Dec/2020].

[47] “Clasificación ROC y AUC” [online] Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419 [Accessed: 15/Dec/2020].

[48] “PlantPepDB. A Manually Curated Plant Peptide Database.” [online] Available: http://223.31.159.8/PlantPepDB/pages/bro.php [Accessed: 09/Dec/2020]

[49] Zhang, L., Ai, H., Chen, W. *et al.* CarcinoPred-EL: Novel models for predicting the carcinogenicity of chemicals using molecular fingerprints and ensemble learning methods. *Sci Rep* **7,** 2118 (2017). https://doi.org/10.1038/s41598-017-02365-0.

[50] Sosnin S, Karlov D, Tetko IV, Fedorov MV. Comparative Study of Multitask Toxicity Modeling on a Broad Chemical Space. J Chem Inf Model. 2019 Mar 25;59(3):1062-1072. doi: 10.1021/acs.jcim.8b00685. Epub 2019 Jan 23. PMID: 30589269.

[51] “Ensemble Learning to Improve Machine Learning Results”. Vadim Smolyakov. [online] Available: https://blog.statsbot.co/ensemble-learning-d1dcd548e936 [Accessed: 04/Jan/2021].

[52] BUENO CRESPO, Andrés, SANCHO GÓMEZ, José Luis. Aprendizaje multitarea en problemas con un número reducido de datos. En: Simposium Nacional de la Unión Científica Internacional de Radio (20º: 2005: Gandia). XX Simposium Nacional de URSI 2005. URSI 05, Gandia del 14-16 de Septiembre, 2005. Valencia: Universidad Politécnica de Valencia, 2005.