Software para predecir la interacción, actividad y función de fármacos

Raúl González Balea
Máster de Bioinformática y Bioestadística
Bioinformática farmacéutica

Melchor Sánchez Martínez
M. Jesús Marco Galindo

05/2016

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Software para predecir la interacción, actividad y función de fármacos*
Nombre del autor:	*Raúl González Balea*
Nombre del consultor/a:	*Melchor Sánchez Martínez*
Nombre del PRA:	*M. Jesús Marco Galindo*
Fecha de entrega (mm/aaaa):	05/2016
Titulación::	*Máster de Bioinformática y Bioestadística*
Área del Trabajo Final:	*Bioinformática farmacéutica*
Idioma del trabajo:	*Castellano*
Palabras clave	*Bases de datos, predicción, similaridad.*

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

Las distintas bases de datos biológicas públicas ofrecen grandes cantidades de información sobre las proteínas y compuestos químicos que interaccionan con ellas: estructura, funcionalidad, propiedades fisicoquímicas, interacciones, etc. En general, la información que contienen se solapa en un 80%, por lo que, si solo se usa una de ellas, un 20% de información se pierde.

Este trabajo unifica la información de muchas de ellas, y crea una matriz que pueda ser usada en procesos de virtual screening y predicción de la interacción target-compuesto químico.

Para ello se han utilizado los datos descargables que ofrecen las distintas páginas web para, utilizando métodos relacionales, cruzarlos. La idea básica es tener una entrada con, al menos, target-compuesto-efectos secundarios. Además, se han añadido campos como los códigos SMILE, InchiKey y propiedades fisicoquímicas del compuesto; secuencia fasta y otros identificadores del target.

Una vez completada, se ha creado un software que, utilizando similaridad de Tanimoto y fingerprints sobre la matriz de datos, dará una lista de compuestos similares y sus posibles efectos secundarios de un fármaco modelo. Aplicando técnicas de machine learning, se ha creado un modelo que trata de predecir si un compuesto interacciona o no con una determinada proteína.

En suma, la matriz de datos proporciona una base fiable y completa de información para que el algoritmo de virtual screening pueda proporcionar resultados satisfactorios. Sin embargo, el algoritmo de machine learning no ofrece conclusiones precisas. No obstante, no se ha podido profundizar mucho en este último punto y requeriría de un estudio más detallado, para solucionar posibles problemas de overfitting.

Abstract (in English, 250 words or less):

Different public data bases offer a big quantity of information about proteins and chemical compounds which interact with: structure, functionality, physicochemical properties, interactions, etc. In general, the information inside overlaps by 80%, so that, if only one is used, 20% of information is lost

This experiment unifies the information of many of them, and creates a matrix that can be used in virtual screening processes and prediction of interaction target-chemical compound.

Therefore downloadable data, offered by the different webs, has been used in order to cross them, using relational methods ,. The aim is to have an entry with, at least, target-compound- side effects . In addition , fields have been added as SMILE code, InchiKey and physicochemical properties of the compound; fasta sequence and other identifiers of target.

Having done that, a software has been created, using Tanimoto similarity and fingerprints on the data matrix, will give a list of similar compounds and their possible side effects of a model drug. Applying machine learning techniques a model has been developed and attempts to predict if a compound interacts with a particular protein or not.

the matrix provides a reliable and comprehensive information base so that the virtual screening algorithm can provide satisfactory results. However, the machine learning algorithm does not provide accurate conclusions. Nevertheless , it has not been able to delve into this , and would require further study to solve possible problems of overfitting.

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 1
1.4 Planificación del Trabajo 2
1.4.1 Planificación inicial 2
1.4.2 Hitos 3
1.4.3 Incidencias y contingencias 4
1.4.4 Materiales 5
1.5 Resumen de productos obtenidos 5
1.6 Descripción de los principales capítulos 5
2. Matriz de datos. 6
2.1 Bases de datos 6
2.2 Matriz de datos 7
3 Virtual Screening 9
3.1 Fingerprint molecular 9
3.2 Similaridad de Tanimoto 11
3.3 Solución con Virtual Screening 11
3.4 Banco de pruebas Virtual Screening 11
4 Análisis SAR con descriptores moleculares 12
4.1 Machine Learning 12
4.2 SVM 13
4.3 Decoys 14
4.4 Fine tunning 15
4.5 Test de precisión y pruebas funcionales. 15
5. Conclusiones 17
5.1 Seguimiento de la planificación 18
5.2 Líneas de trabajo futuras 18
5.2.1 Matriz de datos 19
5.2.2 Integración en servicio Web 19
5.2.3 Theano 19
6. Glosario 21
7. Bibliografía 23
8. Anexos 26
8.1 Diagrama de Gantt 26
8.2 Ejemplo de la matriz de datos 27
8.3 Código Virtual Screening con similaridad de Tanimoto 28
8.4 Ejemplo de prueba funcional Tanimoto-01 29
8.5 Ejemplo de prueba funcional Tanimoto-02 30
8.6 Código predictivo basado en SVM 32
8.7 Ejemplo de prueba funcional ML-01 36
8.8 Ejemplo de prueba funcional ML-02 37

Índice de Ilustraciones

Ilustración 1: Acoplamiento molecular[1] 9
Ilustración 2: Circular fingerprint[2] 10
Ilustración 3: Efecto del radio en fingerprint de Morgan[3] 10
Ilustración 4: Concepto de perceptron(35) 13
Ilustración 5: Hiperplano óptimo vs generalización del hiperplano óptimo (37) 13
Ilustración 6: Influencia del parametro C sobre la precisión del modelo(35) 14
Ilustración 7: Diagrama de Gantt 26

Índice de Tablas

Tabla 1: Hitos del TFM 4
Tabla 2: Incidencias y contingencias 4
Tabla 3: Bases de datos utilizadas 7
Tabla 4: Prueba funcional Tanimoto-01 11
Tabla 5: Prueba funcional Tanimoto-02 12
Tabla 6: Test del modelo de predicción con SVM 16
Tabla 7:Prueba funcional ML-01 16
Tabla 8:Prueba funcional ML-02 16
Tabla 9: Comparativa procesamiento vs coste 20

Índice de ejemplos de código

Código 1: Ejemplo de parser en Python para KEGG 8
Código 2: Optimización automática de SVM 15

1. Introducción

1.1 Contexto y justificación del Trabajo

Cada año se generan miles de millones de bytes de información, a medida que las técnicas de investigación, como las técnicas de secuenciación o los métodos de determinación de la estructura de proteínas, avanzan. La cantidad de datos que almacenan las distintas bases de datos repartidas por todo el mundo está en el orden de los petabytes (10^{15}) (1).

El sector farmacológico no es una excepción, la cantidad de información genómica y proteómica de las distintas enfermedades, así como de la interacción de los fármacos con estas, requiere de técnicas *Big Data* para estudiar e interpretar la misma.

Este trabajo versa sobre la interacción, actividad y función de fármacos con las proteínas que causan determinadas enfermedades. A partir de algunas de las bases de datos biológicas públicas, se genera una matriz de datos con información compuesto-proteína, para evitar la pérdida de información por el escaso solapamiento entre las bases de datos, que, habitualmente, no llega al 80% (2).

A continuación, con técnicas como virtual screening o algoritmos machine learning, se proveen herramientas para la investigación de nuevos fármacos, a partir de la matriz de datos.

1.2 Objetivos del Trabajo

El presente trabajo busca la consecución de los siguientes objetivos:

- Familiarizarse con las principales bases de datos de targets y compuestos.
- Crear una matriz con la relación target-compuesto-efectos secundarios.
- Realizar un algoritmo de virtual screening basado en similaridad.
- Realizar un algoritmo, utilizando machine learning y propiedades fisicoquímicas para predecir la actividad de un compuesto contra un target determinado. En este punto se profundizará en función del tiempo disponible al terminar el resto de tareas, por lo que se considera un objetivo opcional del trabajo.

1.3 Enfoque y método seguido

Para realizar el trabajo, se valoraron dos posibles métodos: por un lado, implementación de una herramienta basada en virtual screening y machine learning. Por otro, utilizar CytoScape, y realizar el mismo análisis mediante redes farmacológicas.

Como tal, no se ha tenido un criterio técnico para seleccionar uno u otro método, ya que ambos tienen una probada eficiencia en estudios de interacción target-compuesto químico. La elección se ha basado en la preferencia del campo de estudio por parte del autor.

1.4 Planificación del Trabajo

Se describen las tareas en las que se divide el trabajo, junto con la definición de hitos y objetivos intermedios. En el anexo 8.1 se puede ver un diagrama de Gantt con la planificación en el tiempo. Además, se identifican las posibles contingencias que se pueden encontrar y se provee una lista de materiales necesarios.

1.4.1 Planificación inicial

A continuación se detallan las tareas definidas en el plan de trabajo inicial:
a/ Exploración y búsqueda de las bases de datos. Matriz de datos.

- *Búsqueda de bibliografía*: se buscará bibliografía relacionada con todos los aspectos clave del proyecto.
- *Leer bibliografía*: se leerá, detenidamente, tanto la bibliografía recomendada por el tutor como la encontrada en el punto anterior
- *Búsqueda de bases de datos y familiarización*: se buscarán bases de datos que contengan información sobre interacciones Target - Compuesto Químico - Efectos Secundarios, familiarizándose con el método de obtención de datos, formatos, fecha de actualización, etc.. Igualmente, se buscará información relevante para la utilización de la librería *Pandas*.
- *Recopilación de datos de las bases de datos seleccionadas*: se obtendrá la información relevante para el trabajo, buscando, en la medida de lo posible, que el formato se adecue a las necesidades.
- *Preparación de formato de las bases de datos*: se formatearán las bases de datos para igualarlas y poder realizar su cruce.
- *Creación de matriz de datos con Pandas*: mediante la librería *Pandas* se cruzarán los datos para formar una matriz.

b/ Pec 2

- *Redactar resultados (PEC2)*:se redactarán y maquetarán los resultados obtenidos hasta la fecha. Se actualizará el plan de trabajo, evaluando la posible desviación de este.
- *Corregir borrador*: revisión sintáctica, semántica y ortográfica de la memoria parcial.

c/ Búsqueda de similaridades

- *Familiarizarse con la métrica de similaridades*: se buscará información sobre la métrica de similaridades en compuestos químicos.
- *Explorar librería RDKIT*: se revisará la información de la librería para utilizar, el Índice de Tanimoto en la búsqueda de similaridades de los compuestos químicos.
- *Implementación de búsqueda por similaridad*: se utilizará la librería anterior para implementar la búsqueda en la matriz de las bases de datos.

d/ Modelo predictivo (opcional)

- *Implementación de modelo Machine Learning*: utilizando la librería scikit-learn, se implementará un algoritmo SVM (Support Vector Machine) predictivo.

- *Testeo de software y correcciones*: se utilizará una molécula con interacciones conocidas para probar el software de predicción. Igualmente, se valorará la creación de un banco de pruebas.

e/ Pec 3

- *Redactar resultados (PEC3)*:se redactarán y maquetarán los resultados obtenidos hasta la fecha. Se actualizará el plan de trabajo, evaluando la posible desviación de este.
- *Corregir borrador*: revisión sintáctica, semántica y ortográfica de la memoria parcial.

f/ Memoria final

- *Líneas de trabajo futuras*: identificar las posibles líneas de continuación del trabajo.
- *Redacción de conclusiones*: redactar conclusiones finales del trabajo.
- *Incorporar PECs a la memoria:* incorporar y adaptar las entregas parciales a la memoria principal.
- *Maquetación:* maquetar memoria.
- *Revisión de la memoria completa*: revisión sintáctica, semántica y ortográfica de la memoria.

g/ Presentación y autoevaluación

- *Selección de contenido*: selección del contenido para incluir en la presentación.
- *Creación de contenido multimedia*: creación de capturas de escritorio para mostrar el software. Generación de gráficos de ayuda.
- *Redacción de la presentación*: redactar presentación con herramienta de diseño tipo PowerPoint o similar. Igualmente, se utilizará un software de grabación de escritorio tipo Camtasia, en caso necesario.
- *Revisión de la presentación*: revisión sintáctica, semántica y ortográfica de la presentación.
- *Realizar informe de autoevaluación*: realizar ejercicio de autocrítica, estimar cumplimiento de la planificación. Rellenar informe con datos.

h/ Implementación de posibles mejoras/Reserva de horas

Esta tarea ha sido introducida en la planificación y cumple dos cometidos:

Debido a que gran parte del trabajo incluye conocimientos que se deben adquirir "sobre la marcha", las horas dedicadas a las tareas de implementación han sido asignadas de forma estimativa. Aunque se tratará de ceñirse a la planificación, cabe la posibilidad de tener que modificarla posteriormente, por lo tanto se han dejado una serie de horas "en reserva" para poder mover la planificación en caso necesario.

1.4.2 Hitos

Los hitos se corresponden con las entregas programadas (resaltadas en verde) y los objetivos marcados en el trabajo.

Tabla 1: Hitos del TFM

Hitos	**Fecha**
Inicio del proyecto	14 de marzo
Matriz de datos	2 de abril
Entrega de la PEC2	4 de abril
Algoritmo de búsqueda finalizado	18 de abril
Entrega de la PEC3	2 de mayo
Software predictivo completo	22 de mayo
Entrega de la Memoria final	30 de mayo
Entrega de la presentación y autoevaluación	6 de junio

1.4.3 Incidencias y contingencias

Se detallan las posibles incidencias que pueden surgir a lo largo de la realización del TFM. Igualmente, se proponen medidas de contingencia para reducir el impacto en la planificación.

Tabla 2: Incidencias y contingencias

Incidencia	**Contingencia**
Avería del puesto de trabajo o corrupción de datos.	– Se dispone de 2 puestos de trabajo adicionales. – Todos los datos del TFM se guardan en servidor con discos en RAID1, así como en local. Igualmente, se mantiene una copia actualizada en cuenta de GMAIL.
Enfermedad o accidente imprevisto.	– Se solicitarán días de vacaciones para recuperar las horas perdidas.
Entregas de otras asignaturas, sin llegar a solaparse, en una misma semana.	– Se solicitarán días de vacaciones si fuese preciso invertir más horas en el TFM.
Puede ocurrir un desplazamiento imprevisto por motivos personales de duración indeterminada.	– Se dispone de puesto de trabajo portátil con copia de la máquina virtual para mantener la planificación en cualquier lugar de residencia temporal.
Desajuste entre planificación y carga real de trabajo.	– En la medida de lo posible se tratará de seguir la planificación. En caso de producirse un descuadre, se asignarán horas adicionales a las tareas con desfase, de la reserva de horas. En caso de no necesitarse, se desarrollarán los objetivos adicionales propuestos.

1.4.4 Materiales

Los materiales que se prevén necesarios para la realización del TFM son:
a/ Software:

- Microsoft Office 2007
- MS Project 2007
- Navegador Google Chrome
- Oracle VM Virtual Box
- Virtualización Ubuntu 32 bits
- Python 2.7 con librerías de análisis de datos (pandas), numéricas (numpy) y machine learning (scikit learn).
- Software Camtasia
- Windows 7

b/ Hardware:

- Punto de trabajo estándar de la UOC (PC de sobremesa)
- Servidor con discos duros en RAID1

1.5 Resumen de productos obtenidos

Los productos obtenidos a la finalización del trabajo son:

- Matriz de datos.
- Script, en Python, que realiza virtual screening basado en similaridad de Tanimoto.
- Script, en Python, que realiza predicción target-compuesto basado en SVM.

1.6 Descripción de los principales capítulos

La memoria final del trabajo consta de los siguientes capítulos:

- Matriz de datos: se describe el proceso de selección, formateo y cruce de las bases de datos de targets y compuestos.
- Virtual screening: se realiza una breve reseña sobre la técnica de virtual screening para, a continuación, presentar los fingerprints y la fórmula de cálculo de similaridad de Tanimoto. Por último, se especifican las pruebas utilizadas en el software y los resultados obtenidos en estas.
- Análisis SAR con descriptores moleculares: se hace una introducción a las técnicas de machine learning y, en especial, las máquinas de soporte vectorial o SVM (del acrónimo inglés, Support Vector Machine). Se detalla la técnica de "ajuste fino" (del inglés, Fine-tuning) y el uso y obtención de decoys. Para finalizar, se definen los test utilizados para valorar el rendimiento del modelo, y se describen las pruebas y los resultados obtenidos.

- Conclusiones: se exponen las conclusiones del trabajo, se analiza el seguimiento de la planificación original y, para concluir, se plantean las posibles líneas futuras de trabajo.
- Bibliografía: se ofrece información sobre la bibliografía utilizada en la elaboración del trabajo.
- Glosario: definición de términos y acrónimos empleados en el presente documento.
- Anexos: contiene pruebas de ejecución y el código de las aplicaciones realizadas.

2. Matriz de datos.

La primera parte de este trabajo ha sido la exploración de las bases de datos de proteínas y compuestos químicos más relevantes que hay en el panorama actual. Cada una de ellas, aporta un ingente número de entradas y referencias a los distintos tipos de targets y/o compuestos; así como interacciones entre ambos. Una vez seleccionadas, se procede a cruzarlas y así generar una matriz de datos lo más completa posible.

2.1 Bases de datos

Dentro de las bases de datos relacionadas con la farmacología se encuentran dos tipos: bases de datos de proteínas (target) y bases de datos de compuestos químicos.
En las primeras, se encuentra toda la información referente a una proteína como puede ser el nombre, secuencia fasta, organismo, transcritos, localización dentro del genoma, y un sin fin de datos adicionales, tanto estructurales como funcionales.
Las segundas, proporcionan datos sobre la química molecular del compuesto, efectos secundarios, propiedades fisicoquímicas, formulación, targets contra los que actúa, SMILES, etc..

Ambos tipos tienen en común algunas características, pero en general, todas utilizan identificadores propios para referirse a los elementos, lo cual muchas veces dificulta el cruzamiento. No obstante, muchas incluyen referencias cruzadas, con lo que se pueden encontrar sinónimos entre los distintos identificadores.
Por otro lado, la mayor parte de ellas proporcionan un API basado en consultas por URL para poder obtener datos directamente desde un script o programa, o si se prefiere, se pueden descargar completamente desde sus páginas web.

El método utilizado en este trabajo ha sido descargar las bases de datos al completo. La siguiente tabla muestra las bases de datos junto con el formato, el tamaño y el tipo de cada una de ellas.

Tabla 3: Bases de datos utilizadas

Base de datos	Formato	Tamaño total de ficheros	Tipo
BioGrid(3)	txt tabulado	3.080 MB	Compuestos
ChEMBL v21	SQL dump	8.272 MB	Compuestos
DisgeNet(4)	txt	201 MB	Targets
DrugBank(5)	xml	391 MB	Targets
ENSEMBL	txt tabulado	241 MB	Targets
KEGG(6)(7)	-	33 MB	Targets
Matador(8)	tsv	2 MB	Compuestos
Reactome(9)(10)	SQL dump	971 MB	Targets
Stitch(11)	csv	11.812MB	Compuestos/Targets
Sider(12)(13)	tsv	641 MB	Compuestos
Uniprot(14)	csv	170MB	Targets

Como se puede ver hay una variedad muy grande de tamaño y formato entre ellas. Siendo algunas tan pequeñas como Matador de apenas 2 MB y otras inmensas como ChEMBL de más de 8GB, que incluye la base de datos tal cual la ofrecen en la web. Los formatos también varían mucho de unas a otras, aunque la mayoría utiliza un formato de campos separados por tabuladores o algún otro carácter.

Durante la fase de selección, se descartaron las siguientes bases de datos: String, OMIM, ClinVar, PubChem, Adrreports y dbSNP. En la mayoría de los casos, la información que contienen es redundante con las demas, o no tienen información útil para este trabajo, como por ejemplo String que contiene interacciones proteína-proteína. DrugBank fue descartada posteriormente por problemas de uso a lo largo del trabajo, los motivos y soluciones que se probaron se detallan en el capítulo 5 Conclusiones.

2.2 Matriz de datos

La creación de la matriz de datos ha sido una de las partes fundamentales del trabajo. Antes de empezar a cruzar las bases de datos, se ha realizado un estudio del formato y del contenido de cada una de ellas, con el objeto de valorar la necesidad de realizar un parser si no tiene un formato organizado por columnas.

Debido al tamaño de los archivos, y pesar de que inicialmente se había propuesto utilizar la librería Pandas para Python, finalmente se ha optado por montar un servidor MySQL y utilizar consultas para el cruce de los datos.

Las bases de datos como Reactome o ChEMBL, que disponen de un volcado en SQL, se han instalado directamente en el servidor. El resto se han parseado, bien con comandos awk, o con scripts en Python, ambos basados en expresiones regulares.

Código 1: Ejemplo de parser en Python para KEGG

```
import re
from __builtin__ import False
fname = "/media/sf_Bases_de_datos/KEGG/disease"

with open(fname) as f:
    content = f.readlines()

f=open('/media/sf_Bases_de_datos/KEGG/matriz_disease.txt','w')
genes = False

for i in content:
    if(i[:8]=="CATEGORY"):
        m = re.search('CATEGORY\s* ([A-Za-z\;\s]*)', i)
        categoria = m.group(1).strip()
        genes = False
        continue
    if(i[:4]=="NAME"):
        m = re.search('NAME\s* ([A-Za-z\s\-]*)', i)
        nombre = m.group(1).strip()
        genes = False
        continue
    if(i[:4]=="GENE"):
        m = re.search('GENE\s* ([A-Za-z0-9\s\-\(\)\/\,\;]*)', i)
        gen =  m.group(1).strip()
        f.write(' '.join([categoria,nombre, gen]))
        f.write("\n")
        genes=True
        continue
    if(i[:6]!="      "):
        genes=False
        continue
    if(genes==True):
        m = re.search('\s* ([A-Za-z0-9\s\-\(\)\/\,\;]*)', i)
        gen =  m.group(1).strip()
        f.write('\t'.join([categoria,nombre, gen]))
        f.write("\n")
        continue
f.close()
```

La selección de los campos se ha realizado siempre con el objetivo de proporcionar la mayor cantidad de información posible de la relación target-compuesto químico.

Respecto del target:

- Identificador ENSEMBL.
- Identificador Uniprot.
- Secuencia fasta.

Respecto del compuesto químico:

- Identificador ChEMBL.
- InchKey.
- SMILE.
- Propiedades fisicoquímicas.
- Efectos secundarios.
-

En el anexo 8.2 hay una ejemplo del formato de la matriz de datos.

3 Virtual Screening

Una de las fases del desarrollo de nuevos fármacos es identificar nuevos compuestos que interacción con un target receptor. En consecuencia, la práctica de probar un gran número de moléculas contra un estructura receptora es un ejercicio común en la empresa farmacéutica(15).

Ahora bien, simular el acoplamiento de una proteína y de cada uno de los posibles ligandos requiere mucho tiempo de computación y, además, una base de datos de estructuras de proteínas de gran tamaño (16). Es por eso que, basándose en la idea de que estructuras similares tienen funciones análogas, se realizan búsquedas de similaridad para encontrar posibles candidatos con estructura semejante a un ligando ya conocido (17).

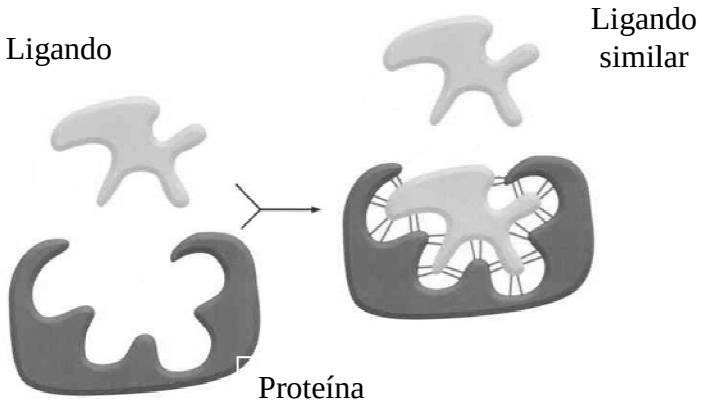

Ilustración 1: Acoplamiento molecular[1]

Apoyándose en esta premisa, se ha realizado un algoritmo que toma un compuesto a estudiar y obtiene su coeficiente de Tanimoto con la matriz de datos, de forma que se obtiene un listado con los 5 compuestos cuya estructura es más similar al compuesto en estudio.

3.1 Fingerprint molecular

Un fingerprint es una abstracción o simplificación que se ha de llevar a cabo de cara a realizar un cálculo de similaridad. Dado que esta puede ser algo subjetiva, sin esta simplificación los resultados tendrían muchas formas de poder interpretarse y, además, el cálculo sería muy complejo (18).

Esta técnica permite comparar moléculas utilizando métodos de comparación de strings binarios. Para ello, las moléculas se transforman generando patrones que se transforman a formato SMILE. A continuación se calcula un hash, y posteriormente se pasa a código binario.

[1]Alberts, B., *et al.* (2008). *Molecular Biology of the Cell (5th ed).* Garland Science: New York.

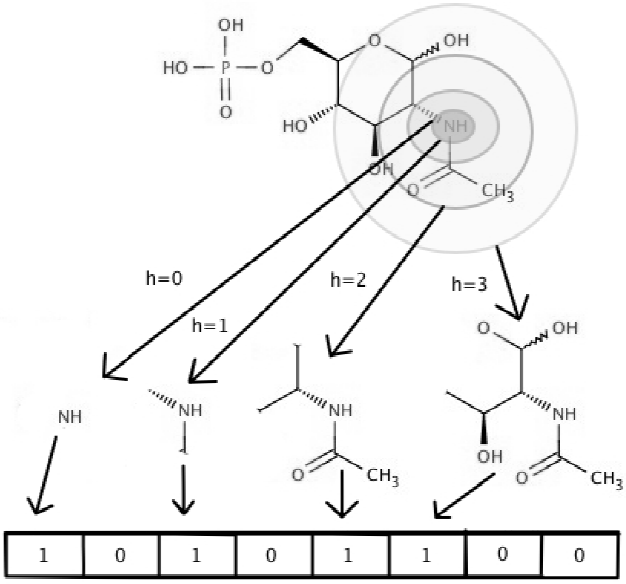

Ilustración 2: Circular fingerprint[2]

Como en el caso de el cálculo de similaridad, existen muchos métodos de llevar a cabo esta simplificación, y también muchos estudios dedicados a analizar y seleccionar los más adecuados para las distintas aplicaciones(19)(18). Se han utilizado estos estudios como fundamento para seleccionar el método de fingerprint para este trabajo, ya que no está en el alcance del mismo realizar una comparativa.

La técnica utilizada ha sido ECFP del tipo fingerprint circular, incluida en la librería RDKIT para Python. Es un método diseñado para el modelado de estructura-actividad basado en el algoritmo de Morgan, y que tiene un gran rendimiento en la búsqueda de similaridad para estructuras (18)(20)(21). Como todos los métodos circulares, se tiene en cuenta el entorno alrededor de cada átomo para obtener los patrones, por ello se le debe de dar un radio indica el tamaño del entorno (18).

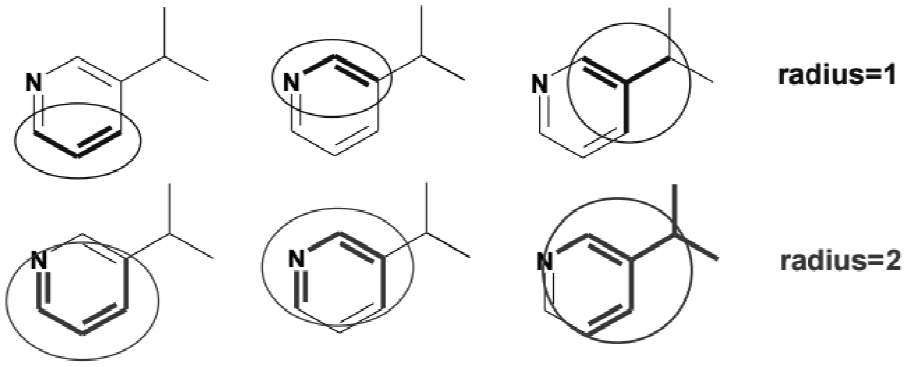

Ilustración 3: Efecto del radio en fingerprint de Morgan[3]

[2] https://chembioinfo.com/2011/10/30/revisiting-molecular-hashed-fingerprints/
[3] http://rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf

3.2 Similaridad de Tanimoto

Dentro de la bioinformática, y también de la quimioinformática, cuantificar la similaridad de dos compuestos es una tarea básica y en cierta medida hasta rutinaria (22). Para ello, existen muchos métodos para realizar los cálculos, entre distintos modos de representación molecular, y un número muy amplio de modos de comparar estas representaciones. Por otro lado, también hay múltiples aplicaciones, casi todas relacionadas con la investigación química en medicina, como por ejemplo: virtual screening (23).

Se han realizado muchos estudios en los cuales se comparan estos métodos, y se trata de seleccionar el mejor en cuanto a eficiencia en distintos escenarios (19)(24)(25)(26).

Para la realización del algoritmo de búsqueda por similaridad, se ha seleccionado el índice de Tanimoto como medida de distancia, ya que a pesar de tener ciertos inconvenientes(27)(28)(29), es uno de los métodos más fiables para obtener la puntuación de similaridad sobre fingerprints de compuestos químicos (30)(31).

El coeficiente de Tanimoto se calcula a partir de la siguiente fórmula:

$$S_{A,B} = \frac{\sum_{j=1}^{n} x_{Aj} x_{Bj}}{\sum_{j=1}^{n} \left(x_{Aj}\right)^2 + \sum_{j=1}^{n} \left(x_{Bj}\right)^2 - \sum_{j=1}^{n} x_{Aj} x_{Bj}}$$

3.3 Solución con Virtual Screening

La librería RDKIT para Python tiene diversas funciones para calcular la similaridad de dos compuestos, entre ellas el coeficiente de Tanimoto. A su vez se han utilizado las funciones de obtención de fingerprint y cálculo de moléculas a partir del código SMILE, también de RDKIT.

El script calcula la distancia entre el compuesto en estudio y todos los compuestos de la matriz, creando fingerprints y calculando el índice de Tanimoto para cada par. Finalmente, el script muestra por pantalla los 10 mejores resultados. El código completo puede verse en el anexo 8.3.

3.4 Banco de pruebas Virtual Screening

Se han realizado las siguientes pruebas clasificadas con nombre Tanimoto-XX:

Tabla 4: Prueba funcional Tanimoto-01

Prueba	Tanimoto-01
Tipo	Funcional

Descripción	Análisis de compuesto sin representación en la matriz
Configuración	Se selecciona un compuesto que no está en la matriz de datos y se pasa a la aplicación.
Resultado esperado	La aplicación devuelve listado con los compuestos más similares en función del índice de Tanimoto, en orden decreciente.
Resultado obtenido	Listado con compuestos similares, en orden decreciente de similaridad Ejemplo de salida en el Anexo 8.4.
Acciones correctivas	No son necesarias

Tabla 5: Prueba funcional Tanimoto-02

Prueba	Tanimoto-02
Tipo	Funcional
Descripción	Análisis de compuesto con representación en la matriz
Configuración	Se selecciona un compuesto que está en la matriz de datos y se pasa a la aplicación.
Resultado esperado	La aplicación devuelve listado con, al menos, el primer puesto con similaridad de valor 1, y resto de listado en función del índice de Tanimoto, en orden decreciente.
Resultado obtenido	Listado con compuestos similares, empezando por el compuesto buscado y similaridad igual 1. Ejemplo de salida en el Anexo 8.5.
Acciones correctivas	No son necesarias

4 Análisis SAR con descriptores moleculares

A medida que las bases de datos de proteínas y compuestos crecen, se ha hecho más necesario refinar la búsqueda de targets, yendo más allá de las técnicas de fuerza bruta. El análisis SAR es uno de los métodos utilizados en el desarrollo de nuevos fármacos y, a la vez, uno de los desafíos más importantes en el campo de los sistemas inteligentes (32).

Existen varias aproximaciones a este problema, como por ejemplo el uso de algoritmos genéticos, redes neuronales o algoritmos de machine learning. Los dos primeros, aunque obtienen resultados excelentes, conllevan algunos problemas asociados con el sobreajuste de los datos de entrenamiento o la reproducibilidad de los resultados (33). Por ende, se ha elegido utilizar machine learning para la parte de análisis SAR del trabajo.

4.1 Machine Learning

Desde el principio machine learning se ha intentado basar en el funcionamiento biológico del cerebro humano, de hecho, lo primero que se hizo fue modelar el comportamiento de una neurona(34). Más adelante, se publicó el primer concepto de

perceptron, un modelo de aprendizaje automático que permite realizar tareas de clasificación. Consta de una función de activación ϕ(z), una combinación lineal de valores de entrada x y un vector de pesos w, siendo z la llamada red de entrada (net input) (35). El funcionamiento de este perceptron es el que se ve en la siguiente figura, siendo la función de activación la que clasifica, entre las diferentes clases, el valor de entrada.

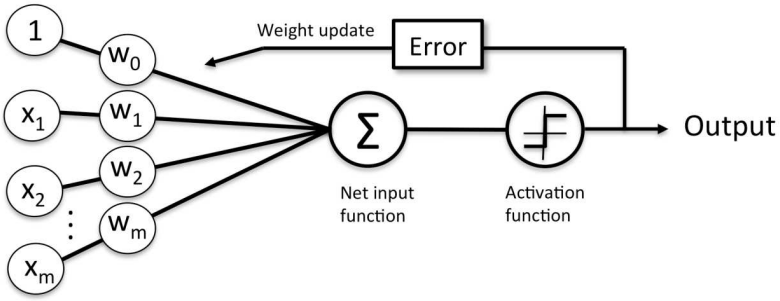

Ilustración 4: Concepto de perceptron(35)

Todos los métodos de machine learning requieren que el modelo sea entrenado previamente hasta alcanzar un nivel de convergencia. A partir de ahí, el modelo ya es capaz de clasificar por sí mismo.
El principal problema del perceptron es que si las clases no son linealmente separables, y además de forma muy extrema, el modelo nunca converge. De ahí la necesidad de desarrollar otros algoritmos diferentes (32).

4.2 SVM

Maquinas de soporte vectorial (SVM, en lo sucesivo) son un grupo de algoritmos de aprendizaje con supervisión, que pueden utilizarse tanto para clasificación como para regresión. Están basados en el perceptron pero, a diferencia de este, en SVM lo que se intenta es maximizar el margen de distancia entre las muestras de entrenamiento y el límite de decisión(35). El elemento límite que se emplea para la separación es un hiperplano y, con su optimización, se pretende que todas las muestras pertenecientes a una clase queden a un lado, y las otras, al otro, maximizando el margen(36).

Como esta optimización no siempre es posible, se asigna un valor llamado coste de clasificación errónea C, con lo que el problema pasa por maximizar el margen de distancia, minimizando el coste (37).

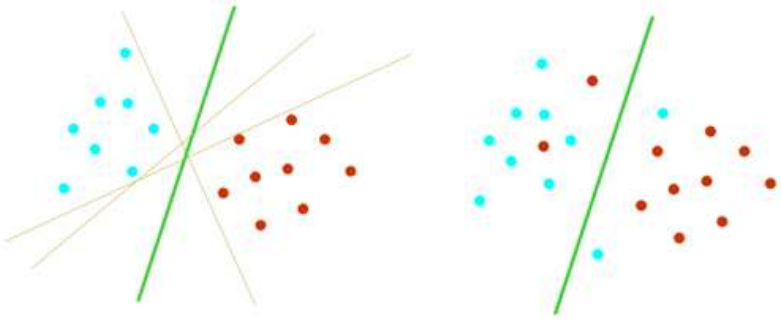

Ilustración 5: Hiperplano óptimo vs generalización del hiperplano óptimo (37)

SVM utiliza diversos kernels, que pueden seleccionarse y ajustarse en función de la distribución de los datos con los que se vaya a trabajar. A estos hay que añadir el valor de coste, que también debe seleccionarse con precisión. En función de cómo se seleccionan el conjunto de parámetros, el modelo puede ser más o menos complejo (35).

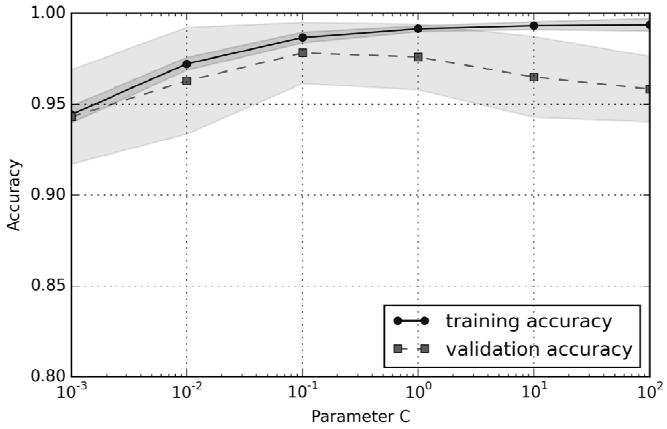

Ilustración 6: Influencia del parametro C sobre la precisión del modelo(35)

En realidad, las imprecisiones del modelo pueden solucionarse aumentando las muestras de entrenamiento, sin embargo, la mayoría de las veces esto puede ser muy costoso, o directamente, imposible (38). De ahí la importancia de seleccionar el tipo de kernel y ajustar los parámetros minuciosamente.

No obstante, los datos que se utilicen tanto para el entrenamiento del modelo como para las predicciones también tienen mucha relevancia. Las propiedades fisicoquímicas seleccionadas se basan en estudios previos, donde se demuestran como las más eficientes para discriminar compuestos (39)(40). Estas son:

- Área de la superficie polar.
- Hidrofobicidad.
- Número de hidrógenos donadores.
- Número de anillos aromáticos.
- Número de anillos alifáticos.
- Índice J de Balaban.

4.3 Decoys

Como en cualquier modelo de aprendizaje es necesario proporcionar ejemplos de qué está bien y qué está mal, de cara a saber tomar decisiones ante casos similares(41). Debido a la inherente falta de control sobre todos los datos de la matriz, es importante controlar y diferenciar los datos que solo aportan ruido, de los datos que aportan ejemplos incorrectos (42).

Para cumplir con esta premisa, en los scripts en los que se utilizan modelos de machine learning se han generado decoys exclusivos para cada target.

Estos se añaden como compuestos inactivos en los dataset de entrenamiento y prueba. Una vez incorporados, se mezclan todos los compuestos de forma aleatoria para no incurrir en un posible error de sobrestimación del rendimiento del modelo (43).

Los decoys se han obtenido de la aplicación web DUD-E(44) en un único archivo con formato .ism, para cada target analizado. En un inicio, se valoró la posibilidad de utilizar Decoyfinder, pero el equipo donde se realizaron las pruebas no soportaba la última versión del programa.

4.4 Fine tunning

El código presente a continuación muestra cómo ha sido realizado el ajuste fino de los parámetros del modelo SVM, utilizando la librería scikit-learn. En él se prueban los siguientes kernels: lineal, polinómico y rbf. La elección de estos tres tipos se basa en que los tres generan un desempeño medio óptimo y, además, tienen un uso muy extendido y probada eficiencia en otros estudios similares(38)(45).

Código 2: Optimización automática de SVM

```
pipe_svc = Pipeline([('scl', StandardScaler()),
            ('clf', SVC(random_state=1))])

param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]

param_grid = [{'clf__C': param_range,
               'clf__kernel': ['linear']},
                {'clf__C': param_range,
                 'clf__gamma': param_range,
                 'clf__kernel': ['rbf']},
                 {'clf__C': param_range,
                  'clf__degree':[1,2,3,4,5],
                  'clf__kernel':['poly']}]

gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',n_jobs=1)

gs = gs.fit(dataset_test, target_test)
```

4.5 Test de precisión y pruebas funcionales.

Utilizando el dataset de test, se realiza una predicción sobre cada compuesto. Cada resultado correcto se suma y se obtiene la precisión del modelo:

$$Accuracy = \frac{\sum correct\ prediction}{\sum total\ compounds} * 100$$

Además, se calculan los errores de tipo I y II; y el estimador F1. Cada medidor se ha tomado tres veces y obtenido la media. Se han utilizado targets con, al menos, 35 compuestos activos.

Tabla 6: Test del modelo de predicción con SVM

Target	Compuestos	Decoys	Precisión	Falso Positivo	Falso negativo	F1 Score
P08684	451	11800	96.77551020408163	1	62	0.67132867132867136
P11712	253	7450	96.29870129870129	2	42	0.6428571428571429
P08588	92	15850	99.40401505646173	0	12	0.8571428571428571
P00797	80	6958	98.08102345415777	2	16	0.66666666666666663
P00918	59	31172	99.83989753442202	0	8	0.46153846153846151
P09874	43	30050	99.91691591890994	0	5	0.61538461538461531
P27338	39	6900	99.56741167988464	0	6	0.57142857142857151
P29274	36	31550	99.88918790565143	0	6	0.61538461538461531
P35354	129	23150	99.54887218045113	0	17	0.63157894736842102
P07550	103	15000	99.47019867549669	0	13	0.72727272727272729

Para comprobar el funcionamiento del programa se realizan las siguientes pruebas funcionales con código ML-XX.

Tabla 7:Prueba funcional ML-01

Prueba	ML-01
Tipo	Funcional
Descripción	Predicción de un compuesto para un target determinado
Configuración	Se obtiene de la matriz de datos los compuestos para el target P08684. Se generan decoys en DUD-E para los compuestos. Se crea un fichero con 5 compuestos activos no presentes en la matriz y 6 compuestos inactivos.
Resultado esperado	La aplicación marca los 5 compuestos activos como efectivos (salida igual a 1) y los 6 compuestos inactivo como inefectivos (Salida igual a 0).
Resultado obtenido	La herramienta predice correctamente 10 de los 11 compuestos. Ejemplo de salida en el Anexo 8.7, el compuesto resaltado en rojo no es correcto.
Acciones correctivas	Los compuestos inactivos tienen una estructura muy parecida a los usados como decoy. El compuesto que no ha sabido identificar correctamente, es muy diferente. Se propone la prueba funcional ML-02

Tabla 8:Prueba funcional ML-02

Prueba	ML-02
Tipo	Funcional
Descripción	Predicción de un compuesto para un target determinado
Configuración	Se obtiene de la matriz de datos los compuestos para el target P08684. Se generan decoys en DUD-E para los compuestos. Se crea un fichero con 5 compuestos activos no presentes en la matriz y 5 compuestos inactivos, todos ellos con estructuras diferentes.
Resultado esperado	La aplicación marca los 5 compuestos activos como efectivos (salida igual a 1) y los 5 compuestos inactivo como inefectivos (Salida igual a 0). Ejemplo de salida en el Anexo 8.8
Resultado obtenido	La herramienta predice correctamente 1 de los compuestos inactivos y 3 de los activos
Acciones correctivas	Analizar posible overfitting o problemas con el número de compuestos con propiedades fisicoquímicas similares.

5. Conclusiones

Una vez concluido el proceso de creación de la matriz de datos, y realizados los algoritmos de virtual screening y predicción, los objetivos propuestos en el plan de trabajo han sido alcanzados.

La principal motivación para realizar este trabajo, ha sido aprender sobre las aplicaciones de la bioinformática en el campo farmacéutico y conocer la metodología utilizada en este tipo de proyectos. Este objetivo se ha cumplido sobradamente.

El proceso de creación de la matriz de datos, aunque no ha estado exento de contratiempos, ha permitido valorar la función que realiza la comunidad científica en el mantenimiento y actualización, de las bases de datos de compuestos y proteínas. Por otro lado, el resultado obtenido solo representa una parte del inmenso trabajo que se podría llegar hacer, para unificar la información de todas y cada una de las bases de datos utilizadas.

El algoritmo de virtual screening funciona perfectamente sobre la matriz de datos, y permite realizar un cribado de compuestos antes de pasar a hacer otro tipo de análisis más exhaustivos. Sin ser un algoritmo muy complejo en cuanto a desarrollo, se han tenido que asimilar conceptos como similaridad estructural, fingerprint y el cálculo de distancias de Tanimoto, para entender el funcionamiento global del mismo.

En contraposición, el algoritmo basado en maquinas de soporte vectorial requiere de una dedicación mayor que la que se le ha podido dar en este trabajo. Se podría considerar que el problema que sufre el modelo es overfitting, es decir, se ajusta perfectamente a los datos de entrenamiento, pero no es capaz de generalizar un modelo para datos reales.
A pesar de haber utilizado técnicas de estandarización y de análisis por componentes para abordar el problema, no se ha experimentado mejoría en la clasificación. Esto puede significar que el problema este en los datos.
El dataset utilizado contiene 451 compuestos activos y 11800 decoys, sin embargo, las moléculas utilizadas pueden no ser semejantes -en cuanto a sus propiedades fisicoquímicas- a las moléculas que se pretenden predecir.
Aun así, la solución aportada puede utilizarse contra cualquier conjunto de datos, correctamente seleccionados, simplemente cambiando los descriptores que se calculan (añadiendo nuevos o modificando los existentes), o seleccionando un mayor número de fármacos para evitar falta de similaridad con los compuestos a predecir.
En definitiva, el trabajo ha permitido adentrarse en una gran cantidad de campos nuevos de estudio, donde todavía falta mucho por investigar, y desarrollar.

5.1 Seguimiento de la planificación

Al elaborar la planificación se cometió un error al subestimar el apartado de creación de la matriz de datos. Inicialmente, se pensó que sería mucho más sencillo el cruce de las bases de datos y, a medida que se iban añadiendo, se encontraron bastantes problemas que implicaron cambios en la planificación. A continuación se detallan las incidencias encontradas durante el trabajo, recopiladas por tarea según la planificación inicial.

a/ Recopilación de las bases de datos:
No se hizo una correcta previsión del espacio que ocupan algunas de las bases de datos, por lo que la máquina virtual se quedó sin espació en disco. La solución aplicada consistió en realizar un snapshot de la máquina, crear una nueva con una capacidad de disco duro mayor, y restaurarla.

b/ Preparación del formato de las bases de datos:
Debido a la complejidad del fichero XML de DrugBank, se realizaron varios intentos de transformarla a un formato más accesible: Se probó la librería SAX de python para convertirla en formato JSON, pero el programa no fue capaz de hacer el parser, debido al anidamiento de etiquetas y al tamaño del fichero. Posteriormente, se probó un parser, hecho en javascript, también sin éxito. También se trató de instalar una base de datos Mongo, ya que DrugBank ofrece fragmentos de la base de datos en formato sdf (Structure Data File). Sin embargo, al importar el fichero, el software Mongo daba un error que no se pudo solventar. Finalmente, se optó por descartarla al tener ya mucha información respecto a compuestos con Stitch y ChEMBLE.

c/ Creación de matriz de datos con Pandas:
Debido al tamaño de las bases de datos, fue imposible utilizar Pandas para cruzarlas: al tener que cargar en memoria todo el fichero, el sistema se bloqueaba. Se valoró el segmentar las bases de datos, pero debiendo mezclarse todos los segmentos entre ellos, el tiempo de ejecución aumentaba drásticamente. Como solución, se optó por instalar un servidor MySQL y volcar las bases de datos en ella, haciendo los cruces mediante consultas SQL en lugar de con Pandas.

5.2 Líneas de trabajo futuras

A pesar de haber alcanzado los objetivos inicialmente planteados, durante el desarrollo del trabajo han surgido momentos en los que se ha dejado de profundizar en algún aspecto, para no salirse del alcance inicial de este. Por otro lado, este trabajo permite utilizarse como base inicial para otros trabajos posteriores.

5.2.1 Matriz de datos

Si bien la matriz de datos alcanzada en el trabajo es óptima y cumple con los requerimientos generales, se podría llegar a crear una base de datos mucho más compleja.

Durante el trabajo, las diferentes bases de datos se han cruzado seleccionando un campo en común entre las que conforman la matriz y las que se van a añadir. En este proceso se pierde información, ya que para evitar campos nulos se ha realizado siempre con inner join, lo que descarta cualquier entrada que no aparezca en ambas. Por consiguiente, si se realiza con outer join las entradas no se pierden, pero hay que completar todos los campos nulos.
Para ello se podría realizar un script que buscase concordancia entre las dos bases de datos, probando diferentes campos: SMILE, InchiKey, secuencia fasta del target, sinónimos de identificadores del compuesto, o del target; y posiblemente otros muchos que están disponibles. Además, no se ha de limitar solo a cruzar bases de datos, se puede utilizar el API que ofrecen muchas de ellas.

Por otro lado, la creación de una base de datos con todos los posibles sinónimos es una tarea que puede llevar mucho tiempo, pero resultaría muy productiva a la larga. En la realización de este trabajo, se ha comprobado que uno de los principales inconvenientes es que muchas de las bases de datos utilizan identificadores propios, por lo que crear una tabla de correspondencia podría ser una gran ayuda para la comunidad.

5.2.2 Integración en servicio Web

La comunidad bioinformática y la mayor parte de entidades del sector ofrecen todas, o al menos gran parte, de sus aplicaciones como servicios web. Sin ir más lejos, herramientas como Blast, Clustal, Swiss Model, Phyre, y muchas otras, están abiertas al uso tanto educativo como profesional. Por añadidura, las bases de datos utilizadas en este trabajo son públicas.

Por consiguiente, se podría realizar un pequeño servidor web basado en HTML y, utilizando Flask, integrar el software creado junto con un buscador para la matriz de datos. La ventaja de Flask es que, al estar escrito en Python, se complementa perfectamente con el código creado.

5.2.3 Theano

Durante la parte de desarrollo del trabajo uno de los inconvenientes no previstos ha sido el tiempo de computación. Aunque en la mayoría de los casos el tiempo total está en torno a 10 minutos, en otros, como por ejemplo el target P08684, el tiempo de ejecución ha llegado a ser superior a 60 minutos.
Investigando a cerca de cómo Python utiliza el procesador de la máquina, se ha visto que debido al GIL (Global Interpreter Lock), Python solo utiliza un núcleo a la vez (35).

Para sacar provecho de equipos con más núcleos existe una librería, llamada multiprocessing, que permite utilizarlos a la vez. Sin embargo, los equipos de escritorio raramente cuentan con más de 8 núcleos, en cambio no es extraño encontrar una tarjeta de video (GPU) con una capacidad de procesamiento, mucho más elevada que la del propio procesador.

Tabla 9: Comparativa procesamiento vs coste

Especificaciones	Intel^{R} Core^{TM} I7-5960X Processor Extreme Edition	NVIDIA GeForce^{R} GTX^{TM} 980 Ti
Frecuencia base del reloj	3.0 GHz	1.0 GHz
Núcleos	8	2816
Ancho de banda de la memoria	68 GB/s	336.5 GB/s
Cálculos en coma flotante	354 GFLOPS	5632 GFLOPS
Coste	1160€	735€

Para aprovechar esta capacidad de procesamiento se desarrolló Theano. Se trata de un lenguaje de programación, un compilador y una librería de Python; todo a la vez, que evalúa expresiones matemáticas con gran eficiencia. Su principal característica es que, además de poder ejecutar código en el procesador, se puede seleccionar la GPU para procesar las expresiones. A modo de ejemplo, en 2010 se realizó un estudio con él y demostró ser hasta 11 veces más rápido que NumPy (46).

6. Glosario

Algoritmo genético: algoritmos basados en el proceso genético de los organismos vivos, que son capaces de ir creando soluciones que evolucionan hacia valores óptimos del problema

Anillo alifático: anillo cíclico de carbono e hidrógeno cuyo carácter no es aromático

Anillo aromático: anillo cíclico de seis átomos de carbono unidos entre sí alternadamente por enlaces dobles y simples deslocalizados

Área de la superficie polar: polar surface area (PSA) se define como la suma de las superficies de los átomos polares (habitualmente oxígeno, nitrógeno e hidrógenos enlazados) en una molécula.

awk: lenguaje de programación diseñado para procesar datos basados en texto, ya sean ficheros o flujos de datos

Big Data: sistema que manipula un gran conjunto de datos

Decoy: son cálculos basados en propiedades físicas similares, pero estructuras químicas distintas

Descriptor molecular: propiedades fisicoquímicas y mediciones experimentales de una molécula que pueden ser transformadas en valores numéricos para ser utilizadas en operaciones matemáticas

Efectos secundarios: acción no buscada, y habitualmente nociva, producida por una sustancia (medicamento) al ser administrado.

Error tipo I y II: el error de tipo I o falso positivo; y error de tipo II o falso negativo

Expresiones regulares: secuencias de caracteres que forma un patrón de búsqueda, las cuales son formalizadas por medio de una sintaxis específica

fasta: formato de fichero informático basado en texto, utilizado para representar secuencias de ácidos nucleicos o péptidos, en el que los pares de bases o los aminoácidos se representan usando códigos de una única letra

Fingerprint: el "fingerprint" molecular es la manera de codificar la estructura de una molécula. La forma más habitual es mediante una serie de dígitos binarios (bits) que representan la presencia o ausencia de una estructura particular en la molécula

hash: algoritmo matemático que mapea datos de tamaño arbitrario en una cadena de bits de un tamaño fijo (una función hash)

Hidrofobicidad: capacidad de una sustancia de ser repelida por el agua

Hidrógenos donadores: en los enlaces por puente de hidrógeno se considera "donador" al átomo que "dona" el hidrógeno para formar la unión. Se considera por tanto el número de enlaces nitrógeno-hidrógeno y oxígeno-hidrógeno.

Índice de Tanimoto: también llamado coeficiente de Jaccard, es una métrica para comparar la similaridad y diversidad de conjuntos de pruebas

Índice J de Balaban: es un índice topológico, entendiendo como tal la caracterización de una molécula o un grafo molecular correspondiente a dicha molécula

inner join: tipo de unión en un contexto de bases de datos relacionales

Ligando: molécula capaz de ser reconocida por otra provocando una respuesta biológica

Machine learning: disciplina que trata de que los sistemas aprendan automáticamente, entendiendo como tal la identificación de patrones complejos en millones de datos, realizado mediante algoritmos de revisión de datos y predicción de comportamientos futuros

outer join: tipo de unión en un contexto de bases de datos relacionales

Parser: analizador sintáctico que convierte un texto de entrada en otras estructuras más útiles para el posterior análisis

Perceptron: tipo de red neuronal artificial desarrollado por Frank Rosenblatt, es la unidad básica para formar redes neuronales

Red neuronal: dentro de la Inteligencia Artificial, se trata de crear sistemas de aprendizaje y procesamiento automáticos inspirados en el funcionamiento de los sistemas nerviosos biológicos

Targets: una molécula con una forma específica sobre la que el fármaco actúa

Virtual screening: técnica de computación utilizada principalmente en farmacología para identificar qué ligandos tienen mayor probabilidad de unirse a un target (diana), habitualmente una proteína, en base a sus estructuras biológicas

7. Bibliografía

1. *Biology: The big challenges of big data.* **Marx, V.** 498, 2013, Nature, págs. 255–260. doi:10.1038/498255a.

2. *Mapping Between Databases of Compounds.* **Muresan, S., Sitzmann, M., Southan, C.** 910, 2012, Methods Mol Biol, págs. 145-64. doi: 10.1007/978-1-61779-965-5_8..

3. *Use of the BioGRID Database for Analysis of Yeast Protein and Genetic Interactions.* **Oughtred R, Chatr-Aryamontri A, Breitkreutz BJ, Chang CS, Rust JM, Theesfeld CL, Heinicke S, Breitkreutz A, Chen D, Hirschman J, Kolas N, Livstone MS, Nixon J, O'Donnell L, Ramage L, Winter A, Reguly T, Sellam A, Stark C, Boucher L, Dolinski K, Tyers M.** 1, 2016, Cold Spring Harb Protoc. doi: 10.1101/pdb.prot088880..

4. *DisGeNET: a discovery platform for the dynamical exploration of human diseases and their genes.* **Piñero, J., Queralt-Rosinach, N., Bravo, À., Deu-Pons, J., Bauer-Mehren, A., Baron, M., Sanz, F., Furlong, LI.** 2015, Database (Oxford). doi: 10.1093/database/bav028.

5. *DrugBank: a comprehensive resource for in silico drug discovery and exploration.* **Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J.** 34, 2006, Nucleic Acids Res., págs. D668-72.

6. *KEGG: Kyoto Encyclopedia of Genes and Genomes.* **Kanehisa, M. and Goto, S.** 28, 2000, Nucleic Acids Res., págs. 27-30.

7. *KEGG as a reference resource for gene and protein annotation.* **Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M., and Tanabe, M.** 44, 2016, Nucleic Acids Res., págs. D457-D462.

8. *SuperTarget and Matador: resources for exploring drug-target relationships.* **Günther S, Kuhn M, Dunkel M, Campillos M, Senger C, Petsalaki E, Ahmed J, Urdiales EG, Gewiess A, Jensen LJ, Schneider R, Skoblo R, Russell RB, Bourne PE, Bork P, Preissner R.** 36, 2008, Nucleic Acids Res., págs. D919-22.

9. *The Reactome pathway Knowledgebase.* **Fabregat, A. Sidiropoulos, K., Garapati, P., Gillespie, M., Hausmann, K., Haw, R., Jassal, B., Jupe, S., Korninger, F., McKay, S., Matthews, L., May, B., Milacic, M., Rothfels, K., Shamovsky, V., Webber, M., Weiser, J., Williams, M., Wu, G., Stein, L.** 44, 2016, Nucleic Acids Res., págs. D481-7. doi: 10.1093/nar/gkv1351.

10. *The Reactome pathway knowledgebase.* **Fabregat, A., Sidiropoulos, K., Garapati, P., Gillespie, M., Hausmann, K., Haw, R., Jassal, B., Jupe, S., Korninger, F., McKay, S., Matthews, L., May, B., Milacic, M., Rothfels, K., Shamovsky, V., Webber, M., Weiser, J., Williams, M., Wu, G., Stein, L.** 42, 2014, págs. D472-7. doi: 10.1093/nar/gkt1102..

11. *STITCH 4: integration of protein-chemical interactions with user data.* **Kuhn, M., Szklarczyk, D., Pletscher-Frankild, S., Blicher, TH., von Mering, C., Jensen, LJ., Bork, P.** 42, 2014, Nucleic Acids Res., págs. 401-407.

12. *The SIDER database of drugs and side effects.* **Kuhn M, Letunic I, Jensen LJ, Bork P.** 2015, Nucleic Acids Res. doi: 10.1093/nar/gkv1075.

13. *A side effect resource to capture phenotypic effects of drugs.* **Kuhn M, Campillos M, Letunic I, Jensen LJ, Bork P.** 2010, Mol Syst Biol., pág. 6:343.

14. *UniProt: a hub for protein information.* **The UniProt Consortium.** 43, 2015, Nucl. Acids Res, págs. D204-D212. doi: 10.1093/nar/gku989.

15. *A geometric approach to macromolecule-ligand interactions.* **Kuntz ID, Blany JM, Oatley SJ, Langridge R, Ferrin TE.** 1982, J Mol Biol., Vol. 161, págs. 269–288. doi:10.1016/0022-2836(82)90153-X.

16. *Methods for Similarity-based Virtual Screening.* **Thomas G. Kristensen, Jesper Nielsen, Christian N. S. Pedersen.** 5, 2013, Computational and Structure Biotechnology Journal, págs. 1-6. doi:10.5936/csbj.201302009.
17. *Similarity-based virtual screening using 2D fingerprints.* **P., Willett.** 11, 2006, Drug Discov Today, págs. 1046-1053.
18. *Molecular Fingerprint Similarity Search in Virtual Screening.* **Cereto-Massagué A, Ojeda MJ, Valls C, Mulero M, Garcia-Vallvé S, Pujadas G.** 71, 2014, Methods, págs. 58-63. doi: 10.1016/j.ymeth.2014.08.005..
19. *How similar are similarity searching methods?: a principal component analysis of molecular descriptor space.* **Bender A, Jenkins JL, Scheiber J, Sukuru SCK, Glick M, Davies JW.** 49, 2009, J Chem Inf Model, págs. 108–119. doi: 10.1021/ci800249s.
20. *Extended-connectivity fingerprints.,.* **D. Rogers, M. Hahn.** 50, 2010, J. Chem. Inf. Model., págs. 742-754. DOI: 10.1021/c160017a018.
21. *The Generation of a Unique Machine Description for Chemical Structures-A.* **Morgan., H.L.** 5, 1965, J. Chem. Doc., págs. 107-113. doi:10.1021/c160017a018..
22. *Molecular similarity: a key technique in molecular informatics.* **Bender A, Glen RC.** 2, 2004, Org Biomol Chem., págs. 3204-3218. doi: 10.1039/b409813g.
23. *Molecular similarity analysis in virtual screening: foundations, limitations and novel approaches.* **Eckert H, Bajorath J.** 12, 2007, Drug Discov Today., págs. 225–3. doi: 10.1016/j.drudis.2007.01.011..
24. *Extended-connectivity fingerprints.* **Rogers D, Hahn M.** 50 , 2010, J Chem Inf Model, págs. 742-754. doi: 10.1021/ci100050t..
25. *Impact of similarity measures on web-page clustering.* **Strehl A, Strehl E, Ghosh J, Mooney R.** [ed.] AAAI. 2000, Proceedings of the Workshop on Artificial Intelligence for Web Search (AAAI 2000), págs. 58-64.
26. *Benchmarking of multivariate similarity measures for high-content screening fingerprints in phenotypic drug discovery.* **Reisen F, Zhang X, Gabriel D, Selzer P.** 18, 2013, J Biomol Screen., págs. 1284–1297. doi: 10.1177/1087057113501390..
27. *Dissimilarity-based compound selection techniques.* **MS, Lajiness.** Perspectives in Drug Discovery and Design, Vol. 7, págs. 65-84.
28. *On the properties of Bit string-based measures of chemical similarity.* **DR, Flower.** 38, 1998, J Chem Inf Comput Sci., págs. 379-386. doi: 10.1021/ci970437z..
29. *Combinatorial preferences affect molecular similarity/diversity calculations using binary fingerprints and Tanimoto coefficients.* **Godden JW, Xue L, Bajorath J.** 40, 2000, J Chem Inf Comput Sci., págs. 163-166. doi: 10.1021/ci990316u..
30. *Similarity coefficients for binary chemoinformatics data: overview and extended comparison using simulated and real data sets.* **Todeschini R, Consonni V, Xiang H, Holliday J, Buscema M, Willett P.** 52, 2012, J Chem Inf Model., págs. 2884–2901. doi: 10.1021/ci300261r..
31. *Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?* **Dávid Bajusz, Anita Rácz, and Károly Héberger.** 2015, J Cheminform, págs. 7-20. doi: 10.1186/s13321-015-0069-3.
32. *Drug design by machine learning: support vector machines.* **R. Burbidge, M. Trotter, B. Buxton, S. Holden.** 26, 2001, Computers and Chemistry, págs. 5-14.
33. *Neural networks in drug discovery: have they lived up to their promise?* **Manallack, D.T., Livingstone, D.J.** 35, 1999, European Journal of Medicinal Chemistry, págs. 95-208.
34. *A Logical Calculus of the Ideas Immanent in Nervous Activity.* **McCulloch, W.S., Pitts, W.** 5, 1943, The bulletin of mathmatical biophysics, págs. 115-133.
35. **Raschka, S.** *Python Machine Learning.* Birmingham : Packt Publishing, 2015.

36. **Djuris, Jelena.** *Computer-Aided Applications in Pharmaceutical Technology.* Cambridge : Woodhead Publishing Limited, 2013.
37. **Gunn, Steve R.** *Support Vector Machines for Classification and Regression.* Southampton : University of Southampton, 1998.
38. **Kumar Thakur, V., Kumari Thakur, M.** *Handbook of Polymers for Pharmaceutical Technologies, Processing and Applications.* Massachusetts : Scrivener Publishing, 2015.
39. *MLViS: A Web Tool for Machine Learning-Based Virtual Screening in Early-Phase of Drug Discovery and Development.* **Korkmaz, S., Zararsiz, G., Goksuluk, D.** 2015, PLOS ONE. DOI:10.1371/journal.pone.0124600.
40. *DrugLogit: logistic discrimination between drugs and nondrugs including disease-specificity by assigning probabilities based on molecular properties.* **García-Sosa AT, Oja M, Hetényi C, Maran U.** 52, 2012, J Chem Inf Model., págs. 2165–2180. doi: 10.1021/ci200587h.
41. **Myers, David G.** *Exploring Psychology.* New York : Worth Publishers, 1990.
42. *Learning from Decoys to Improve the Sensitivity and Specificity of Proteomics Database Search Results.* **Yadav, A. Kumar, Kumar, D., Dash, D.** 7, 2012. doi:10.1371/journal.pone.0050651.
43. *An Unsupervised, Model-Free, Machine-Learning Combiner for Peptide Identifications from Tandem Mass Spectra.* **Edwards, N., Wu, X., Tseng, C.** 1, 2009, Clinical Proteomics, Vol. 5, págs. 23-36. DOI: 10.1007/s12014-009-9024-5.
44. *Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking.* **Mysinger, MM., Carchia, M., Irwin, JJ., Shoichet, BK.** 14, 2012, J. Med. Chem., Vol. 55, págs. 6582–6594. DOI: 10.1021/jm300687e.
45. *On the parameter optimization of Support Vector Machines fo rbinary classification.* **Gaspar, P., Carbonell, J., Oliveira J. Luis.** 9, 2012, Journal of Integrative Bioinformatics, págs. 201-212.
46. *Theano: A CPU and GPU Math Compiler in Python.* **Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.** 2010, PROC. OF THE 9th PYTHON IN SCIENCE CONF.

8. Anexos

8.1 Diagrama de Gantt

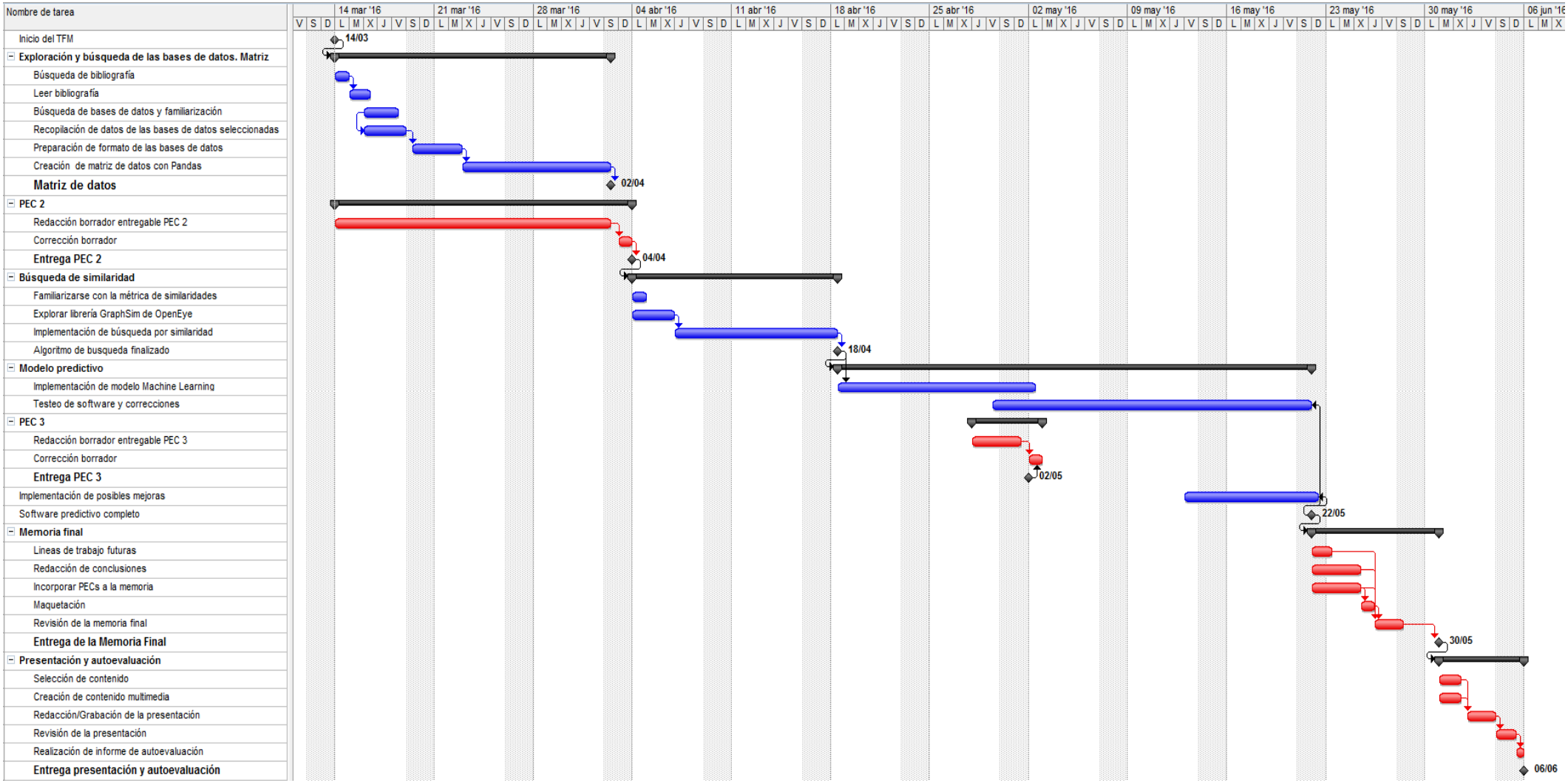

Ilustración 7: Diagrama de Gantt

8.2 Ejemplo de la matriz de datos

ens_target	ENSP	uniprot	chembl_comp	inchi	smile	alogp	rtb	hba	hbd	psa	Side effects
ENSP00000260682	ENSP00000260682	P11712	CHEMBL106	RFHAOTPXVQNOHP-UHFFFAOYSA-N	OC(Cn1cncn1)(Cn2cncn2)c3ccc(F)cc3F	0	5	5	1	81	;;Abnormal vision;Agitation;Anorexia;Aspergillosis;Aspergillus infection;Blood triglycerides increased;Chills;Cross resistance;Decreased appetite;Dermatitis;Discomfort;Erythema multiforme;Feeling abnormal;Fungal infection;Gastrointestinal pain;Hyperhidrosis;Hypopnoea;Ill;Infestation;Infestation NOS;labyrinth disorders;Lacrimation;Lacrimation increased;Leukorrhea;Liver injury;Loose stools;lymphatic system disorders;Menopausal symptoms;Musculoskeletal discomfort;Opportunistic mycosis;Pathogen resistance;Phosphatase alkaline increased;Rigors;Salivary hypersecretion;Salivation;Shallow breathing;Sweating increased;Tension;Unspecified disorder of skin and subcutaneous tissue;Urethral disorder;Urine output increased;Vaginal discharge;Vaginal infection;Vascular purpura;Visual impairment
ENSP00000275016	ENSP00000275016	Q9NYL5	CHEMBL106	RFHAOTPXVQNOHP-UHFFFAOYSA-N	OC(Cn1cncn1)(Cn2cncn2)c3ccc(F)cc3F	0	5	5	1	81	;;Abnormal vision;Agitation;Anorexia;Aspergillosis;Aspergillus infection;Blood triglycerides increased;Chills;Cross resistance;Decreased appetite;Dermatitis;Discomfort;Erythema multiforme;Feeling abnormal;Fungal infection;Gastrointestinal pain;Hyperhidrosis;Hypopnoea;Ill;Infestation;Infestation NOS;labyrinth disorders;Lacrimation;Lacrimation increased;Leukorrhea;Liver injury;Loose stools;lymphatic system disorders;Menopausal symptoms;Musculoskeletal discomfort;Opportunistic mycosis;Pathogen resistance;Phosphatase alkaline increased;Rigors;Salivary hypersecretion;Salivation;Shallow breathing;Sweating increased;Tension;Unspecified disorder of skin and subcutaneous tissue;Urethral disorder;Urine output increased;Vaginal discharge;Vaginal infection;Vascular purpura;Visual impairment
ENSP00000176183	ENSP00000176183	P21917	CHEMBL1026	CVKUMNRCIJMVAR-UHFFFAOYSA-N	CS(=O)(=O)O.Oc1ccc(cc1)C2CNCCc3c(Cl)c(O)c(O)cc23	3	1	4	4	72	;Acute coronary syndrome;Blood creatinine increased;Blood lactate dehydrogenase increased;Cramp of limb;Creatinine increased;Gastrointestinal pain;Haemoglobin;Hyperhidrosis;Lactic dehydrogenase activity increased;Muscle spasms;Sweating;Tension
ENSP00000386069	ENSP00000386069	P18825	CHEMBL1026	CVKUMNRCIJMVAR-UHFFFAOYSA-N	CS(=O)(=O)O.Oc1ccc(cc1)C2CNCCc3c(Cl)c(O)c(O)cc23	3	1	4	4	72	;Acute coronary syndrome;Blood creatinine increased;Blood lactate dehydrogenase increased;Cramp of limb;Creatinine increased;Gastrointestinal pain;Haemoglobin;Hyperhidrosis;Lactic dehydrogenase activity increased;Muscle spasms;Sweating;Tension
ENSP00000260682	ENSP00000260682	P11712	CHEMBL1043	MQJKPEGWNLWLTK-UHFFFAOYSA-N	Nc1ccc(cc1)S(=O)(=O)c2ccc(N)cc2	1	2	4	2	94	;Application site pain;Drug;Gastrointestinal pain;Joint sprain;Ligament sprain;Localized exfoliation;Lupus;Photosensitivity reaction;Phototoxicity;Protein urine present;Skin exfoliation
ENSP00000286479	ENSP00000286479	P11245	CHEMBL1043	MQJKPEGWNLWLTK-UHFFFAOYSA-N	Nc1ccc(cc1)S(=O)(=O)c2ccc(N)cc2	1	2	4	2	94	;Application site pain;Drug;Gastrointestinal pain;Joint sprain;Ligament sprain;Localized exfoliation;Lupus;Photosensitivity reaction;Phototoxicity;Protein urine present;Skin exfoliation
ENSP00000337915	ENSP00000337915	P08684	CHEMBL1043	MQJKPEGWNLWLTK-UHFFFAOYSA-N	Nc1ccc(cc1)S(=O)(=O)c2ccc(N)cc2	1	2	4	2	94	;Application site pain;Drug;Gastrointestinal pain;Joint sprain;Ligament sprain;Localized exfoliation;Lupus;Photosensitivity reaction;Phototoxicity;Protein urine present;Skin exfoliation
ENSP00000360372	ENSP00000360372	P33261	CHEMBL1043	MQJKPEGWNLWLTK-UHFFFAOYSA-N	Nc1ccc(cc1)S(=O)(=O)c2ccc(N)cc2	1	2	4	2	94	;Application site pain;Drug;Gastrointestinal pain;Joint sprain;Ligament sprain;Localized exfoliation;Lupus;Photosensitivity reaction;Phototoxicity;Protein urine present;Skin exfoliation

8.3 Código Virtual Screening con similaridad de Tanimoto

```
import pandas as pd
import numpy as np
from rdkit import Chem, DataStructs
from rdkit.Chem import AllChem
import sys
'''
Obtain the key for sorting array
'''
def getKey(item):
    return item[0]
'''
Tanimoto's similarity search
'''
def similarityTest(ifile, compound):
    try:
        print "Now working on dataset: ",ifile
        print "Compound to search: ",compound
        df = pd.read_table(ifile, sep=",")

        mol1   =   Chem.MolFromSmiles(compound)#   Compound   to   search
similarity
        fp1 = AllChem.GetMorganFingerprintAsBitVect(mol1,2,2048)
        col=df['smile']
        best_tani = 0
        init = 0
        array_best = []# Array with the best results
        print "Calculating Tanimoto similarity...",
        for idx,val in enumerate(col):
            mol2 = Chem.MolFromSmiles(val)
            fp2 = AllChem.GetMorganFingerprintAsBitVect(mol2,2,2048)
            tani = DataStructs.TanimotoSimilarity(fp1, fp2)
            if(init <10): # Filling the array with initial compounds
                array_best.append([np.float(tani), idx])
                array_best.sort(key=getKey, reverse=True)
                init = init + 1
                best_tani = array_best[-1][0]
            if(tani>=best_tani and init == 10): # Selecting the best
similarity compounds
                array_best[-1]=[tani,idx]
                array_best.sort(key=getKey, reverse=True)
                best_tani = array_best[-1][0]
        print "Finished"
        return array_best,df
    except IOError:
        print "Can not open ",ifile
        print "Please check file's path"
        exit(-1)
    except:
        print("Unexpected error:", sys.exc_info()[0])
        raise
'''
Main
'''
if __name__ == '__main__':

array_best,df=similarityTest("/media/sf_Bases_de_datos/Matriz_compuest
os.txt", "OC(Cn1cncn1)(Cn2cncn2)c3ccc(F)cc3F")
    '''
    Printing results
```

```
    '''
    print "Printing results..."
    print ""
    for i in array_best:
        print(df.iloc[i[1]] ,i[0])
        print""
```

8.4 Ejemplo de prueba funcional Tanimoto-01

```
Now working on dataset:  /media/sf_Bases_de_datos/Matriz_compuestos.txt
Compound to search:  CCC[C@@H](O)CC\C=C\C=C\C#CC#C\C=C\CO
Calculating Tanimoto similarity... Finished
Printing results...

(chembl_comp                                              CHEMBL109
inchi                                  NIJJYAXOARWZEE-UHFFFAOYSA-N
smile                                             CCCC(CCC)C(=O)O
alogp                                                           2
rtb                                                             5
hbd                                                             1
hba                                                             2
psa                                                            37
Side           ;;Abdominal cramps;Abnormal vision;Abscess per...
Name: 10, dtype: object, 0.1951219512195122)

(chembl_comp                                              CHEMBL433
inchi                                  AEQFSUDEHCCHBT-UHFFFAOYSA-M
smile                                     [Na+].CCCC(CCC)C(=O)[O-]
alogp                                                           2
rtb                                                             5
hbd                                                             1
hba                                                             2
psa                                                            37
Side           ;;Abdominal cramps;Abnormal vision;Abscess per...
Name: 192, dtype: object, 0.18181818181818182)

(chembl_comp                                         CHEMBL2105613
inchi                                  MSRILKIQRXUYCT-UHFFFAOYSA-M
smile                  [Na+].CCCC(CCC)C(=O)O.CCCC(CCC)C(=O)[O-]
alogp                                                           2
rtb                                                             5
hbd                                                             1
hba                                                             2
psa                                                            37
Side           ;;Abdominal cramps;Abnormal vision;Abscess per...
Name: 169, dtype: object, 0.16666666666666666)

(chembl_comp                                         CHEMBL1200896
inchi                                  IYGXEHDCSOYNKY-RZHHZEQLSA-N
smile          CCCCC[C@H](O)C=C[C@H]1[C@H](O)C[C@H](O)[C@@H]1...
alogp                                                           2
rtb                                                            12
hbd                                                             4
hba                                                             5
psa                                                            97
Side           ;Agitation;Dermatitis;Discomfort;Drowsiness;Ex...
Name: 888, dtype: object, 0.16)

(chembl_comp                                              CHEMBL548
inchi                                  XEYBRNLFEZDVAW-ARSRFYASSA-N
smile          CCCCC[C@H](O)C=C[C@H]1[C@H](O)CC(=O)[C@@H]1CC=...
alogp                                                           3
rtb                                                            12
hbd                                                             3
hba                                                             5
psa                                                            94
Side           ;Acute coronary syndrome;Amniotic fluid emboli...
```

```
Name: 269, dtype: object, 0.15492957746478872)

(chembl_comp                  CHEMBL1200453
inchi          NOPFSRXAKWQILS-UHFFFAOYSA-N
smile              CCCCCCCCCCCCCCCCCCCCCO
alogp                                   9
rtb                                    20
hbd                                     1
hba                                     1
psa                                    20
Side                          ;Dermatitis
Name: 905, dtype: object, 0.15)

(chembl_comp                                            CHEMBL1051
inchi                                 GGXICVAJURFBLW-CEYXHVGTSA-N
smile          CC(C)OC(=O)CCCC=C/C[C@H]1[C@@H](O)C[C@@H](O)[C...
alogp                                                           4
rtb                                                            14
hbd                                                             3
hba                                                             5
psa                                                            86
Side           ;Asthmatic attack;Localised skin reaction;Musc...
Name: 800, dtype: object, 0.14285714285714285)

(chembl_comp                                              CHEMBL448
inchi                                WEXRUCMBJFQVBZ-UHFFFAOYSA-N
smile                              CCCC(C)C1(CC)C(=O)NC(=O)NC1=O
alogp                                                           1
rtb                                                             4
hbd                                                             2
hba                                                             3
psa                                                            75
Side           ;CNS depression NOS;Depressed level of conscio...
Name: 619, dtype: object, 0.13725490196078433)

(chembl_comp                                              CHEMBL962
inchi                                 LMHIPJMTZHDKEW-XQYLJSSYSA-M
smile          [Na+].CCCCC[C@H](O)C=C[C@H]1[C@H](O)C[C@@H]2OC...
alogp                                                           2
rtb                                                            10
hbd                                                             3
hba                                                             5
psa                                                            86
Side           ;Abdominal colic;Abdominal distension;Acute co...
Name: 235, dtype: object, 0.12987012987012986)

(chembl_comp                                            CHEMBL1029
inchi                                 UQRORFVVSGFNRO-UTINFBMNSA-N
smile                    CCCCN1C[C@H](O)[C@@H](O)[C@H](O)[C@H]1CO
alogp                                                           0
rtb                                                             4
hbd                                                             4
hba                                                             5
psa                                                            84
Side           ;Abdominal bloating;Abdominal distension gaseo...
Name: 382, dtype: object, 0.12962962962962962)
```

8.5 Ejemplo de prueba funcional Tanimoto-02

```
Now working on dataset:  /media/sf_Bases_de_datos/Matriz_compuestos.txt
Compound to search:  OC(Cn1cncn1)(Cn2cncn2)c3ccc(F)cc3F
Calculating Tanimoto similarity... Finished
Printing results...

(chembl_comp                                              CHEMBL106
inchi                                 RFHAOTPXVQNOHP-UHFFFAOYSA-N
```

```
smile                          OC(Cn1cncn1)(Cn2cncn2)c3ccc(F)cc3F
alogp                                                          0
rtb                                                            5
hbd                                                            1
hba                                                            5
psa                                                           81
Side           ;;Abnormal vision;Agitation;Anorexia;Aspergill...
Name: 0, dtype: object, 1.0)

(chembl_comp                                            CHEMBL638
inchi                                  BCEHBSKCWLPMDN-MGPLVRAMSA-N
smile            C[C@@H](c1ncncc1F)[C@](O)(Cn2cncn2)c3ccc(F)cc3F
alogp                                                          2
rtb                                                            5
hbd                                                            1
hba                                                            5
psa                                                           76
Side           ;Abdominal distension;Abnormal vision;Acute co...
Name: 713, dtype: object, 0.5849056603773585)

(chembl_comp                                           CHEMBL1399
inchi                                  YBBLVLTVTVSKRW-UHFFFAOYSA-N
smile                   CC(C)(C#N)c1cc(Cn2cncn2)cc(c1)C(C)(C)C#N
alogp                                                          2
rtb                                                            4
hbd                                                            0
hba                                                            4
psa                                                           78
Side           ;Accidental injury;Acute coronary syndrome;Ano...
Name: 109, dtype: object, 0.3018867924528302)

(chembl_comp                                           CHEMBL1397
inchi                                  RAGOYPUPXAKGKH-XAKZXMRKSA-N
smile          CC[C@@H]([C@H](C)O)N1N=CN(C1=O)c2ccc(cc2)N3CCN...
alogp                                                          5
rtb                                                           12
hbd                                                            1
hba                                                            9
psa                                                          111
Side           ;Abdominal pain aggravated;Abnormal platelets;...
Name: 675, dtype: object, 0.2608695652173913)

(chembl_comp                                        CHEMBL1201032
inchi                                  JPRXYLQNJJVCMZ-UHFFFAOYSA-N
smile           CN(C)CCc1c[nH]c2ccc(Cn3cncn3)cc12.OC(=O)c4ccccc4
alogp                                                          2
rtb                                                            5
hbd                                                            1
hba                                                            3
psa                                                           49
Side           ;Dermatitis;Gastrointestinal pain;Hyperhidrosi...
Name: 663, dtype: object, 0.25)

(chembl_comp                                            CHEMBL898
inchi                                    HUPFGZXOMWLGNK-UHFFFAOYSA-N
smile                                OC(=O)c1cc(ccc1O)c2ccc(F)cc2F
alogp                                                          3
rtb                                                            2
hbd                                                            2
hba                                                            3
psa                                                           57
Side           ;;Acute anaphylaxis;Allergic cutaneous angiiti...
Name: 328, dtype: object, 0.2222222222222222)

(chembl_comp                                           CHEMBL1444
inchi                                  HPJKCIUCZWXJDR-UHFFFAOYSA-N
smile                       N#Cc1ccc(cc1)C(c2ccc(cc2)C#N)n3cncn3
alogp                                                          2
```

```
rtb                                                              3
hbd                                                              0
hba                                                              4
psa                                                             78
Side           ;Acute coronary syndrome;Anorexia;Anxiety depr...
Name: 112, dtype: object, 0.20754716981132076)

(chembl_comp                                              CHEMBL407
inchi                                 OFBIFZUFASYYRE-UHFFFAOYSA-N
smile                      CCOC(=O)c1ncn2c1CN(C)C(=O)c3cc(F)ccc23
alogp                                                            1
rtb                                                              3
hbd                                                              0
hba                                                              4
psa                                                             64
Side           ;Abnormal vision;Affect lability;Anxiety attac...
Name: 814, dtype: object, 0.2)

(chembl_comp                                              CHEMBL924
inchi                                 XRASPMIURGNCCH-UHFFFAOYSA-N
smile                            OC(Cn1ccnc1)(P(=O)(O)O)P(=O)(O)O
alogp                                                           -1
rtb                                                              4
hbd                                                              5
hba                                                              8
psa                                                            172
Side           ;;Advanced breast cancer;Anorexia;Anxiety depr...
Name: 769, dtype: object, 0.19230769230769232)

(chembl_comp                                             CHEMBL1306
inchi                                 BLSQLHNBWJLIBQ-OZXSUGGESA-N
smile          CC(C)N1CCN(CC1)c2ccc(OC[C@H]3CO[C@@](Cn4cncn4)...
alogp                                                            4
rtb                                                              8
hbd                                                              0
hba                                                              7
psa                                                             64
Side           ;Dermatitis;Discomfort;Feeling abnormal;Gastro...
Name: 451, dtype: object, 0.18604651162790697)
```

8.6 Código predictivo basado en SVM

```
import numpy as np
import pandas as pd
import time
import sys
from sklearn.svm import SVC
from sklearn import svm
from sklearn.grid_search import GridSearchCV
from sklearn import preprocessing
from sklearn.pipeline import Pipeline
from sklearn import cross_validation
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from rdkit import Chem
from rdkit.Chem import Descriptors



def readDataIntoNp(ifile,dfile):
    try:
        reader = pd.read_table(ifile, sep=",")
        print "Now working on dataset: ",ifile
```

```
        col=reader.as_matrix()
        if not col.all():
            raise ValueError("Check file's format")

        dataset=[]

        '''
        Fill dataset with molecule physicochemical's properties and
label 1
        '''
        print "Creating dataset...",
        for row in col:
            mol=Chem.MolFromSmiles(row[0])

dataset.append(np.array([[Descriptors.MolLogP(mol),Descriptors.NumHDon
ors(mol),Descriptors.TPSA(mol),Descriptors.NumAromaticRings(mol),Descr
iptors.NumAliphaticRings(mol),Descriptors.BalabanJ(mol)],np.int(1)]))
        print "Finished"

        '''
        Adding decoys with molecule physicochemical's properties and
label 0
        '''
        print "Adding decoys...",
        decoys=[]
        suppl = Chem.SmilesMolSupplier(dfile, titleLine=False)

        for mol in suppl:

decoys.append(np.array([[Descriptors.MolLogP(mol),Descriptors.NumHDono
rs(mol),Descriptors.TPSA(mol),Descriptors.NumAromaticRings(mol),Descri
ptors.NumAliphaticRings(mol),Descriptors.BalabanJ(mol)],np.int(0)]))
        print "Finished"

        '''
        Randomizing dataset
        '''
        dataset.extend(decoys)
        dataset= np.array(dataset)
        np.random.shuffle(dataset)

        '''
        Split dataset with proportion 80-20 for train and test
        '''

        print "Generating test and train datasets...",
        target_test = dataset[0:np.int(len(dataset)*0.2),1].tolist()
        dataset_test = dataset[0:np.int(len(dataset)*0.2),0].tolist()
        target_train = dataset[np.int(len(dataset)*0.2):,1].tolist()
        dataset_train = dataset[np.int(len(dataset)*0.2):,0].tolist()
        print "Finished"


        return dataset_train,target_train,dataset_test,target_test

    except IOError:
        print "Can not open ",ifile
        print "Please, check file's path"
        exit(-1)
    except ValueError:
        print "Please, check files'format for errors"
```

```
        raise
        exit(-1)
    except:
        print("Unexpected error:", sys.exc_info()[0])
        raise

if __name__ == '__main__':
    t1=time.time()
    '''
    Generating datasets
    '''

dataset_train,target_train,dataset_test,target_test=readDataIntoNp("/m
edia/sf_Bases_de_datos/Pruebas/p08684/matriz.txt",'/media/sf_Bases_de_
datos/Pruebas/p08684/decoys_final2.ism')

    '''
    Data standarization and fitting the model
    '''
    scaler = preprocessing.StandardScaler().fit(dataset_train)
    dataset_train_std = scaler.transform(dataset_train)
    dataset_test_std = scaler.transform(dataset_test)

    '''
    Hyperparameters optimization
    '''

    pipe_svc    =    Pipeline([('scl',    StandardScaler()),    ('clf',
SVC(random_state=1))])


    param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]

    '''
    Kernels to test
    '''

    param_grid = [{'clf__C': param_range,
                'clf__kernel': ['linear'],
                'clf__class_weight':['balanced', None]},
                  {'clf__C': param_range,
                   'clf__gamma': param_range,
                   'clf__kernel': ['rbf'],
                   'clf__class_weight':['balanced', None]},
                   {'clf__C': param_range,
                    'clf__degree':[1,2,3,4,5,6],
                    'clf__kernel':['poly'],
                    'clf__class_weight':['balanced', None]}]

    gs                                                                  =
GridSearchCV(estimator=pipe_svc,param_grid=param_grid,scoring='accurac
y',n_jobs=1,cv=6)
    print "Starting fine tuning...",

    gs = gs.fit(dataset_train_std, target_train)
    print "Finished "

    print(gs.best_score_)
    print(gs.best_params_)

    '''
```

```
    Construct and fit SVM
    '''
    print "Fitting the model ",
    if(gs.best_params_['clf__kernel']=='rbf'):
        print "rbf...",
        svm    =    SVC(C=gs.best_params_['clf__C'],    cache_size=200,
class_weight=gs.best_params_['clf__class_weight'],
gamma=gs.best_params_['clf__gamma'], kernel='rbf')
    elif(gs.best_params_['clf__kernel']=='linear'):
        print "linear...",
        svm    =    SVC(C=gs.best_params_['clf__C'],    cache_size=200,
class_weight=gs.best_params_['clf__class_weight'], kernel='linear')
    else:
        print "poly...",
        svm    =    SVC(C=gs.best_params_['clf__C'],    cache_size=200,
class_weight=gs.best_params_['clf__class_weight'],
degree=gs.best_params_['clf__degree'], kernel='poly')

    '''
    Fitting the model
    '''
    svm.fit(dataset_train_std, target_train)
    print "Finished "
    t2=time.time()
    print "Computing total time: ",t2-t1
    '''
    Test the model with test datasets
    '''
    print "Testing...",
    scores   =   cross_validation.cross_val_score(svm,   dataset_train,
target_train, cv=5)
    print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()
* 2))


    correct_count = 0 # Number of correct predictions
    positive_false = 0
    negative_false = 0
    target_predicted = [] # Array of predictions for F1 score
    for idx,val in enumerate(dataset_test_std):
        predicted = svm.predict(np.array(val).reshape(1, -1))[0]
        if predicted == target_test[idx]:
            correct_count += 1
        elif predicted == 1 and target_test[idx] == 0:
            positive_false += 1
        else:
            negative_false += 1
        target_predicted.append(predicted)

    print "Finished "
    print("Accuracy:                                                    ",
(np.float(correct_count)/len(dataset_test_std)) * 100.00)
    print("Test    accuracy    :    ",    svm.score(dataset_test_std,
target_test))
    print("Type I error: ", positive_false)
    print("Type II error: ", negative_false)
    print("F1                                                  Score:
",f1_score(target_test,np.array(target_predicted), average='binary'))
    '''
    Prediction
    '''
```

```
    print "Initializing predictions"
    reader                                                          =
pd.read_table("/media/sf_Bases_de_datos/Pruebas/p08684/prediction.txt"
, sep=",")
    col=reader.as_matrix()
    for smile in col:
        mol=Chem.MolFromSmiles(smile[0])
        mol                                                         =
scaler.transform(np.array([Descriptors.MolLogP(mol),Descriptors.NumHDo
nors(mol),Descriptors.TPSA(mol),Descriptors.NumAromaticRings(mol),Desc
riptors.NumAliphaticRings(mol),Descriptors.BalabanJ(mol)]).reshape(1,
-1))
        orig_pp = svm.predict(mol)
        print"Prediction for: ", smile, orig_pp
```

8.7 Ejemplo de prueba funcional ML-01

```
Now working on dataset:  /media/sf_Bases_de_datos/Pruebas/p08684/matriz.txt
Creating dataset... Finished
Adding decoys... Finished
Generating test and train datasets... Finished
Starting fine tuning... Finished
0.914127423823
{'clf__gamma': 1.0, 'clf__C': 10.0, 'clf__class_weight': None, 'clf__kernel':
'rbf'}
Fitting the model  rbf... Finished
Computing total time:  415.93937397
Testing... Accuracy: 0.79 (+/- 0.02)
Finished
('Accuracy: ', 92.22222222222223)
('Test accuracy : ', 0.92222222222222228)
('Type I error: ', 10)
('Type II error: ', 18)
('F1 Score: ', 0.81818181818181823)

Initializing predictions
Prediction                                                          for:
['CCOc1ccc(cc1OC)[C@H]2c3c(=O)c4cc(ccc4oc3C(=O)N2CCc5ccccc5)Cl'] [0]
Prediction                                                          for:
['CCCOc1ccc(cc1)CN([C@@H]2CCS(=O)(=O)C2)C(=O)c3c(c4c(s3)cccc4Cl)Cl'] [0]
Prediction                                                          for:
['Cc1ccc(cc1)c2cnc(o2)CCC(=O)N(Cc3c(cccc3Cl)F)[C@@H]4CCS(=O)(=O)C4'] [0]
Prediction  for:   ['CCc1ccc(cc1)OCCOc2c(cc(cc2Br)/C=C\\3/C(=O)N(C(=S)S3)C)OC']
[1]
Prediction                                                          for:
['C[C@@H](C(=O)OC)Oc1ccc2c(c1)oc(c(c2=O)Oc3ccccc3Br)C(F)(F)F'] [0]
Prediction for:  ['[Cu+2].[O-]S(=O)(=O)[O-]'] [1]
Prediction                          for:                                [
'[H][C@@]12CC[C@@](O)(C#C)[C@@]1(C)CC[C@@]1([H])[C@@]2([H])CCC2=CC3=C(C[C@]12C
)C=NO3'] [1]
Prediction for:  ['CC(=O)NC1=CC=C(O)C=C1'] [1]
Prediction           for:                      ['CC(=O)CC(C1=CC=C(C=C1)[N+]([O-
])=O)C1=C(O)C2=CC=CC=C2OC1=O'] [1]
Prediction                          for:                                [
'C[C@H]1C[C@H]2[C@@H]3CC[C@](O)(C(=O)C)[C@@]3(C)C[C@H](O)[C@]2(F)[C@@]4(C)C=CC
(=O)C=C14'] [1]
Prediction                          for:                                [
'CC(C)c1nc(CN(C)C(=O)N[C@@H](CCN2CCOCC2)C(=O)N[C@H](CC[C@H](Cc3ccccc3)NC(=O)OC
c4cncs4)Cc5ccccc5)cs1'] [1]
```

8.8 Ejemplo de prueba funcional ML-02

```
Now working on dataset:  /media/sf_Bases_de_datos/Pruebas/p08684/matriz.txt
Creating dataset... Finished
Adding decoys... Finished
Generating test and train datasets... Finished
Starting fine tuning... Finished
0.909027777778
{'clf__gamma': 1.0, 'clf__C': 1.0, 'clf__class_weight': None, 'clf__kernel':
'rbf'}
Fitting the model  rbf... Finished
Computing total time:  396.186175108
Testing... Accuracy: 0.79 (+/- 0.01)
Finished
('Accuracy: ', 91.36490250696379)
('Test accuracy : ', 0.91364902506963785)
('Type I error: ', 3)
('Type II error: ', 28)
('F1 Score: ', 0.79738562091503273)
Initializing predictions
Prediction for:  ['[H][C@@](C)(C(O)=O)C1=CC(=CC=C1)C(=O)C1=CC=CC=C1'] [0]
Prediction for:  ['[Cu+2].[O-]S(=O)(=O)[O-]'] [1]
Prediction for:  ['OC[C@@H](O1)[C@@H](O)[C@H](O)[C@@H](O)[C@@H](O)1'] [1]
Prediction for:  ['CCC[C@@H](O)CC\\\\C=C\\\\C=C\\\\C#CC#C\\\\C=C\\\\CO'] [1]
Prediction for:  ['CC(=O)NCCC1=CNc2c1cc(OC)cc2'] [1]
Prediction                    for:                                                [
'CC[C@H](C)C(=O)O[C@H]1C[C@@H](C)C=C2C=C[C@H](C)[C@H](CC[C@@H]3C[C@@H](O)CC(=O
)O3)[C@@H]12'] [1]
Prediction                    for:                                                [
'CC[C@H]1OC(=O)[C@H](C)[C@@H](O[C@H]2C[C@@](C)(OC)[C@@H](O)[C@H](C)O2)[C@H](C)
[C@@H](O[C@@H]3O[C@H](C)C[C@@H]([C@H]3O)N(C)C)[C@](C)(O)C[C@@H](C)CN(C)[C@H](C
)[C@@H](O)[C@]1(C)O'] [1]
Prediction                    for:                                                [
'CC[C@H]1OC(=O)[C@H](C)[C@@H](O[C@H]2C[C@@](C)(OC)[C@@H](O)[C@H](C)O2)[C@H](C)
[C@@H](O[C@@H]3O[C@H](C)C[C@@H]([C@H]3O)N(C)C)[C@](C)(O)C[C@@H](C)C(=O)[C@H](C
)[C@@H](O)[C@]1(C)O'] [1]
Prediction for:  ['OC1(CCN(CCCC(=O)c2ccc(F)cc2)CC1)c3ccc(Cl)cc3'] [0]
Prediction for:  ['CN1C(=C(O)c2ccccc2S1(=O)=O)C(=O)Nc3ncc(C)s3'] [0]
```