



ESCUELA
POLITÉCNICA
SUPERIOR

Análisis de relaciones entre medicamentos



Grado en Ingeniería Biomédica

Trabajo Fin de Grado

Autor:

Sergio Sáez Martínez

Tutor/es:

María Paloma Moreda Pozo

Mayo 2022

Agradecimientos

En primer lugar, agradecer a mi tutora Paloma Moreda su guía durante todo el proyecto, proporcionándome ayuda en todas las dificultades que han ido surgiendo. También a Jose Ignacio Abreu, por hacer más fácil poder entender una tecnología innovadora como el procesamiento del lenguaje natural.

Además, de a mi familia y amigos por apoyarme siempre en todo lo que he necesitado y estar a mi lado.

Gracias a todos.

Resumen

La Ingeniería Biomédica surge para solucionar los problemas existentes en el campo de la medicina y sus ramas, haciendo uso de las nuevas tecnologías que van surgiendo.

Estas nuevas técnicas nos van a permitir tratar problemas en el ámbito de la salud que deberían tenerse más en cuenta, como son las interacciones entre medicamentos. Este es un tema primordial ante el incremento del consumo de medicamentos por parte de la población, puesto que un aumento de las mismas puede derivar en efectos adversos desconocidos. Esto, junto con la poca información que hay sobre ellas, supone una gran amenaza para la salud pública y la industria farmacéutica.

Existen muy pocos estudios relacionados con este problema, y todos ellos se tratan desde un punto de vista bioquímico. Por lo tanto, lo trataremos desde una perspectiva distinta de las que podemos encontrar. Así bien, mediante la semántica de las palabras y el análisis de sentimientos, podremos hallar similitudes entre los medicamentos. Todo ello, aplicando técnicas de procesamiento del lenguaje natural, una tecnología que cada vez cobra más importancia.

El procesamiento del lenguaje natural es según IBM “la rama de la informática -más concretamente, de la inteligencia artificial- que se ocupa de dotar a los ordenadores de la capacidad de entender el lenguaje hablado y escrito del mismo modo que los seres humanos. Esta tecnología ha alcanzado un alto nivel en la actualidad gracias a la aplicación de tecnologías como el *machine learning* (aprendizaje automático), el *big data*, el internet de las cosas o redes neuronales.”

Proponemos un estudio que mediante las diferentes características que podemos detectar de los medicamentos, consigamos un modelo que prediga la relación positiva, negativa o neutra entre ellos.

Con este sistema, intentaremos ayudar a los profesionales a evitar las interacciones peligrosas a la hora de administrar o recetar fármacos en pacientes polimedicados, y a las farmacéuticas a diseñar ensayos clínicos evitando combinaciones nocivas.

Palabras clave

Interacción medicamentos, Reacción adversa, Principio Activo, Procesamiento del lenguaje natural, Análisis de sentimientos

Abstract

Biomedical Engineering arises to solve existing problems in the field of medicine and its fields, making use of new and emerging technologies.

The interactions between medicines is a major issue that should be taken into account due to the raise in the consumption of medicines by the population, since an increase in these interactions can lead to unknown adverse effects. This, combined with the lack of information about them, is a major threat for public health and the pharmaceutical industry.

There are very few studies related to this problem, and all of them are treated from a biochemical point of view. For this reason, we are going to carry out one from a different perspective from those we can find. Thus, by means of word semantics and sentiment analysis, we can find similarities between medicines. All this, by applying natural language processing techniques, a technology that is becoming increasingly important.

According to IBM, natural language processing is "the branch of computer science - more specifically, artificial intelligence - that deals with giving computers the ability to understand spoken and written language in the same way as human beings. This technology has reached a high level today thanks to the application of technologies such as machine learning, big data, the internet of things or neural networks".

We propose a study that, by means of the different characteristics that we can detect of the medicines, we will obtain a model that predicts the positive, negative or neutral relationship between them.

With this system we will try to help professionals to avoid dangerous interactions when administering or prescribing medication to polymedicated patients, and to help pharmaceutical companies to design clinical trials avoiding harmful combinations.

Índice de contenidos

Agradecimientos.....	3
Resumen.....	4
Abstract.....	5
Índice de tablas e ilustraciones.....	7
Abreviaturas.....	9
Introducción.....	10
Objetivos.....	12
Estado del arte.....	13
Metodología.....	18
Desarrollo de la propuesta.....	19
Diseño de la base de datos.....	19
Añadir información a la base de datos.....	22
Actualización de la base de datos.....	25
Desarrollo del modelo.....	26
Caso de uso.....	41
Análisis de costes.....	43
Planificación del proyecto.....	43
Costes.....	44
Conclusiones.....	45
Implementaciones futuras.....	45
Bibliografía.....	46
Anexos.....	49
Anexo1.....	49
Anexo 2.....	50
Anexo 3.....	52
Anexo 4.....	54
Anexo 5.....	57
Anexo 6.....	60
Anexo 7.....	66
Anexo 8.....	71

Índice de tablas e ilustraciones

Tabla 1. Tipos de reacciones adversas. Elaboración propia	13
Tabla 2. Costes software y hardware. Elaboración propia	44
Ilustración 1. Medicamentos vendidos en España.[Gráfica].En “España, cada vez más medicada”, por A. Ordaz, S. Pérez, V. Oliveres , 2022, https://www.eldiario.es/sociedad/espana-vez-medicada_1_8796419.html	10
Ilustración 2. Modelo entidad-relación. Fuente: Elaboración propia	19
Ilustración 3. Atributos de la tabla. Fuente: Elaboración propia	20
Ilustración 4. Añadir claves primarias. Fuente: Elaboración propia	21
Ilustración 5. Referenciar claves ajenas. Fuente: Elaboración propia	21
Ilustración 6. Tablas de la base de datos. Fuente: Elaboración propia	21
Ilustración 7. Insertar medicamentos. Fuente: Elaboración propia	23
Ilustración 8. Medicamentos añadidos a la base de datos. Fuente: Elaboración propia ...	23
Ilustración 9. Principios activos añadidos a la base de datos. Fuente: Elaboración propia	23
Ilustración 10. Composición medicamentos añadidos a la base datos. Fuente: Elaboración propia	24
Ilustración 11. Insertar interacciones. Fuente: Elaboración propia	24
Ilustración 12. Interacciones añadidas a la base de datos. Fuente: Elaboración propia ...	25
Ilustración 13. Actualizar base de datos. Fuente: Elaboración propia	25
Ilustración 14. Diagrama modelo aprendizaje automático. Fuente: Elaboración propia ...	27
Ilustración 15. Librerías. Fuente: Elaboración propia	27
Ilustración 16. Archivo CSV interacciones. Fuente: Elaboración propia	28
Ilustración 17. Código limpiar texto y leer datos. Fuente: Elaboración propia	28
Ilustración 18. Datos limpios. Fuente: Elaboración propia	28
Ilustración 19. Funcionamiento TF-IDF.[Imagen]. En El proceso de cálculo de TF-IDF en TfidfVectorizer en sklearn. Fuente: https://programmerclick.com/article/70301137743/ ...	29
Ilustración 20. TF-IDF configuración. Fuente: Elaboración propia	29
Ilustración 21. Modelos de predicción. Fuente: Elaboración propia	30
Ilustración 22. Score primera prueba. Fuente: Elaboración propia	30
Ilustración 23. F1 Score primera prueba. Fuente: Elaboración propia	30
Ilustración 24. Reporte clasificación primera prueba. Fuente: Elaboración propia	31
Ilustración 25. Confusion matrix primera prueba. Fuente: Elaboración propia	31
Ilustración 26. Prefijos y sufijos útiles en medicamentos. Core Radiológico. (2011). Prefijos y sufijos útiles en medicamentos [Fotografía]. URL: http://www.coreradiologico.com/contenidos/zonas/detalleApartado.aspx?idAPadre=xx0FI7%2FcbVfwa0MvdU%2FWSA%3	32
Ilustración 27. Introducir datos terminaciones y preprocesarlos. Fuente: Elaboración propia	33
Ilustración 28. Archivo CSV interacciones con terminaciones. Fuente: Elaboración propia	33
Ilustración 29. Archivo CSV principio activo-terminaciones. Fuente: Elaboración propia ..	33

Ilustración 30. Score segunda prueba. Fuente: Elaboración propia	33
Ilustración 31. F1 Score segunda prueba. Fuente: Elaboración propia.....	34
Ilustración 32. Reporte de clasificación segunda prueba. Fuente: Elaboración propia.....	34
Ilustración 33. Confusion matrix segunda prueba. Fuente: Elaboración propia.....	34
Ilustración 34. Información sobre principios activos DrugBank. Fuente: https://go.drugbank.com/drugs/DB01048	35
Ilustración 35. Código actualizado preprocesado del texto. Fuente: Elaboración propia..	36
Ilustración 36. Datos limpios tercera prueba. Fuente: Elaboración propia	36
Ilustración 37. Score tercera prueba. Fuente: Elaboración propia	36
Ilustración 38. F1 Score tercera prueba. Fuente: Elaboración propia	36
Ilustración 39. Reporte clasificación tercera prueba. Fuente: Elaboración propia	37
Ilustración 40. Confusion matrix tercera prueba. Fuente: Elaboración propia	37
Ilustración 41. Nuevo principio activo. Fuente: Elaboración propia	38
Ilustración 42. Añadir características del nuevo principio. Fuente: Elaboración propia	38
Ilustración 43. Preprocesar los nuevos datos. Fuente: Elaboración propia	39
Ilustración 44. Preparar datos para la clasificación. Fuente: Elaboración propia	39
Ilustración 45. Tabla para la clasificación. Fuente: Elaboración propia	40
Ilustración 46. Cargar modelos. Fuente: Elaboración propia	40
Ilustración 47. Transformar a vector de características. Fuente: Elaboración propia	40
Ilustración 48. Predecir relaciones del nuevo principio. Fuente: Elaboración propia	40
Ilustración 49. Nuevo medicamento añadido a CIMA. Fuente: Elaboración propia	41
Ilustración 50. Medicamento añadido a la base de datos	41
Ilustración 51. Características del nuevo principio. Fuente: Elaboración propia.....	41
Ilustración 52. Dataset actualizado con nueva información. Fuente: Elaboración propia .	41
Ilustración 53. Resultados de las nuevas interacciones. Fuente: Elaboración propia	42

Abreviaturas

PNL -> Procesamiento del lenguaje natural

RAM -> Reacciones adversas a medicamentos

CIMA -> Centro de información online de medicamentos de la Agencia Española de Medicamentos y Productos Sanitarios

TF-IDF -> *Term Frequency Inverse Document Frequency Vectorizer*

Introducción

En el ámbito médico, buscan y se interesan por prevenir, diagnosticar o tratar una enfermedad. Pero para ello, es necesario escoger los medicamentos que se administraran y llevar una vigilancia y un control de los efectos que estos provocan.

Observamos como año tras año aumenta el consumo de medicamentos en la población. Esto puede llegar a ser un problema debido a que existen personas que se automedican incluso a veces sin ser necesario, se consulta a varios profesionales al mismo tiempo y no se tiene en cuenta que medicamentos tiene recetados.

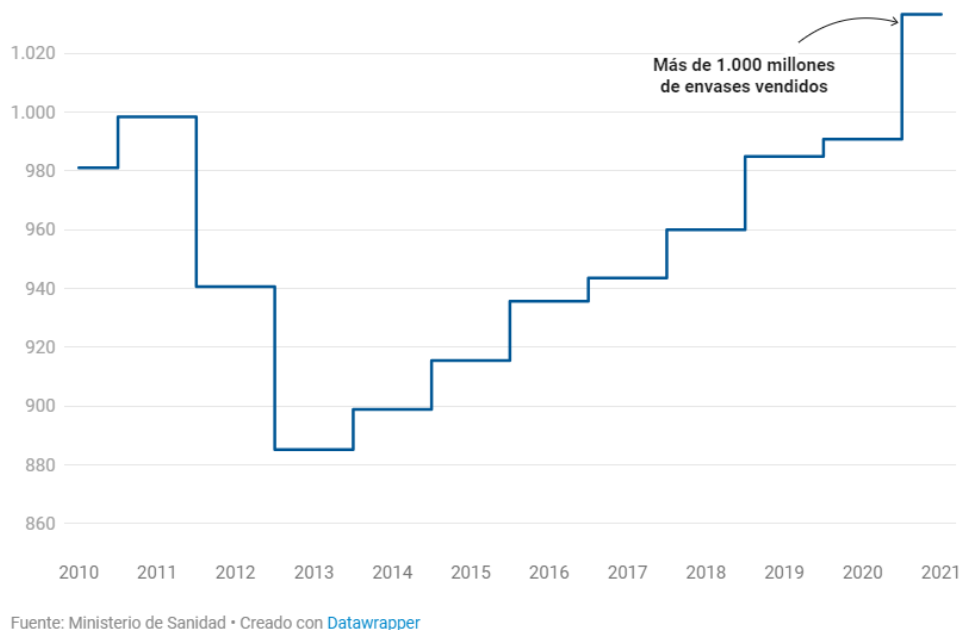


Ilustración 1. Medicamentos vendidos en España.[Gráfica].En “España, cada vez más medicada”, por A. Ordaz, S. Pérez, V. Oliveres , 2022, https://www.eldiario.es/sociedad/espana-vez-medicada_1_8796419.html

La polimedicación puede tener graves consecuencias si se utilizan medicamentos que no deben administrarse al mismo tiempo. Algunos de los efectos que pueden darse son: aumento de efectos secundarios, reacciones adversas o interacciones medicamentosas e intoxicaciones, entre muchos otros factores.

¿Cómo de graves pueden llegar a ser las reacciones adversas y de qué manera se controlan?

Las reacciones adversas a medicamentos (RAM) son una causa muy poco estudiada, que puede ocasionar el ingreso en hospitales o centros de salud, e incluso pueden llegar a casuar la muerte. En ocasiones, pasan desapercibidas por el personal sanitario y se revelan una vez se investiga la causa de la muerte.

En el ámbito de la salud, la función de controlar las reacciones adversas o informar sobre ellas se lleva a cabo mediante la farmacovigilancia. La farmacovigilancia es (OMS, 2002) la ciencia que trata de recoger, vigilar, investigar y evaluar la información sobre los efectos de los medicamentos, productos biológicos, plantas medicinales y medicinas tradicionales, con el objetivo de identificar información sobre nuevas reacciones adversas y prevenir daños en los pacientes.

Esta actividad se lleva a cabo durante el proceso de investigación y desarrollo de los fármacos, ya que es normal que se produzcan reacciones adversas. Algunas de estas reacciones, si son pequeñas y tolerables, se incluirán en la lista de efectos secundarios, pero pueden ocurrir reacciones que impidan que el fármaco sea viable.

Las reacciones adversas que se detectan en los ensayos clínicos son las más comunes ya que se realizan con un número limitado de personas. Las menos comunes se manifiestan una vez el fármaco se comercializa y se administra.

Muchos de los artículos que existen sobre las reacciones adversas o interacciones entre medicamentos, concluyen que estas reacciones se han detectado a posteriori, es decir, una vez el paciente lo ha sufrido o se ha realizado un estudio observando detenidamente el uso de los fármacos en su conjunto.

Por consiguiente, con este proyecto vamos a intentar predecir si las relaciones entre los fármacos serán positivas, negativas o neutras, con el objetivo principal de facilitar la prevención de interacciones entre medicamentos o reacciones adversas.

Objetivos

El objetivo principal del trabajo consiste en analizar las relaciones entre medicamentos y determinar si dichas relaciones tendrán un carácter negativo, positivo o neutro, mediante el uso de técnicas de procesamiento del lenguaje natural.

Para ello será necesario:

- Definir y crear un corpus con las características adecuadas.
- Familiarizarnos con el uso de técnicas de procesamiento del lenguaje natural, técnicas de aprendizaje automático, en concreto, sistemas de análisis de sentimientos.
- Desarrollar un sistema de análisis capaz de predecir el carácter de las relaciones entre medicamentos.

Estado del arte

Anteriormente, se ha mencionado mucho el término “reacción adversa a medicamentos” o “interacción medicamentosa”. A continuación, procederemos a definir y explicar estos términos con el fin de identificarlos en posteriores menciones.

¿Qué son las reacciones adversas y qué tipos existen?

Según la OMS, esta es una “reacción nociva y no deseada que se presenta tras la administración de un medicamento, a dosis utilizadas habitualmente en la especie humana, para prevenir, diagnosticar o tratar una enfermedad, o para modificar cualquier función biológica”.

Existen diversos tipos de reacciones:

Tipos	Descripción
Tipo A (Aumentadas)	Respuesta aumentada de la dosis.
Tipo B (Bizarras)	Reacciones no relacionadas con los efectos del fármaco.
Tipo C (Crónicas)	Reacciones producidas por tratamientos largos.
Tipo D (Diferidas)	Salen después de acabar el tratamiento.
Tipo E (“End”)	Aparecen tras la suspensión repentina del medicamento.
Tipo F (“Failure”)	Reacciones originadas por agentes ajenos a los principios activos.

Tabla 1. Tipos de reacciones adversas. Elaboración propia

¿Qué son las interacciones medicamentosas y qué tipos existen?

Las interacciones medicamentosas o farmacológicas según Wikipedia son “la modificación del efecto de un fármaco por la acción de otro cuando se administran conjuntamente” o en un corto periodo de tiempo.

Pueden duplicar su acción si se administran dos fármacos que tienen el mismo efecto e intensificarse sus reacciones adversas. También oponerse, es decir, si dos fármacos con efectos opuestos reducen la efectividad de uno o los dos. Además, alterar la forma en la que se absorbe o se metaboliza en el cuerpo.

Existen dos tipos de interacciones:

- Interacciones farmacocinéticas: Se producen modificaciones en la concentración de los fármacos. Estas modificaciones se deben a diferencias en la absorción, transporte, distribución, metabolización o excreción de uno o los dos fármacos con respecto a las esperadas de cada fármaco si se toman de forma individual.
- Interacciones farmacodinámicas: Modifica la respuesta del organismo ante la llegada del fármaco.

¿Cómo puede un facultativo o un paciente evitar las interacciones entre medicamentos?

- Evitando la automedicación.
- Reduciendo la administración de medicamentos lo máximo posible.
- Teniendo registrado por parte del médico todos los medicamentos que consume su paciente, y revisando si se puede producir o no interacción.
- Prestando atención a los hábitos dietéticos y al consumo de alcohol.

¿Cómo se indican las interacciones medicamentosas en un medicamento?

En caso de comprar medicamentos sin receta, leer el prospecto del fármaco, puesto que en él se indican posibles efectos adversos y medicamentos que interaccionan con él, aunque no todos. Debemos recordar que los prospectos se actualizan, por tanto es importante leerlo cada vez que se usa un fármaco aunque se haya consumido anteriormente.

El prospecto nos proporciona diferente información sobre el fármaco que hemos adquirido. A continuación, vamos a escoger un fármaco para analizarlo.

Podemos consultar cualquier prospecto en la web de CIMA (Centro de información online de medicamentos de la Agencia Española de Medicamentos y Productos Sanitarios) (<https://cima.aemps.es/>).

Por ejemplo, seleccionamos el prospecto del medicamento Zyvoxid (antibiótico que actúa impidiendo el crecimiento de bacterias) (https://cima.aemps.es/cima/dochtml/p/64107/P_64107.html#1-qu-es-zyvoxid-y-para-qu-se-utiliza).

Las partes de un prospecto son las siguientes:

1. Qué es Zyvoxid y para qué se utiliza:

- Explica qué clase de medicamento es y su uso.

2. Qué necesita saber antes de usar Zyvoxid:

- Advertencias y precauciones.
- Información sobre su uso.
- Interacciones con otros medicamentos.
- Composición y cantidad de cada principio activo que lo compone.

3. Cómo tomar Zyvoxid:

- Cantidad y cada cuánto tiempo hay que tomarlo.

4. Posibles efectos adversos:

- Se indican los posibles efectos adversos que se pueden sufrir y la frecuencia con la que suelen producirse.

5. Conservación de Zyvoxid:

- Cómo conservar el medicamento.

6. Contenido del envase e información adicional:

- Principio activo principal, componentes secundarios y excipientes.
- Información sobre comercialización, farmacéutica y fecha de revisión del prospecto.

Aun así, siempre que sea posible se debe consultar con el médico sobre el medicamento que se va a consumir e informar en el caso de que se estén tomando otros, para evitar posibles interacciones y efectos adversos.

Debemos tener en cuenta que todos los medicamentos están formados por principios activos, que son los causantes de que se produzcan las relaciones y surjan las interacciones que provocan los efectos adversos.

Un principio activo se define según el BOE como “la sustancia o mezcla de sustancias utilizadas para elaborar un medicamento con una acción farmacológica, inmunológica o metabólica para restaurar, corregir o modificar las funciones fisiológicas o con fines diagnósticos”.

Para analizar y predecir las relaciones entre medicamentos, utilizaremos las relaciones entre los principios activos que los forman. Para ello, vamos a utilizar técnicas de procesamiento del lenguaje natural (PLN) y análisis de sentimientos.

¿Qué es el procesamiento del lenguaje natural (PLN)?

El procesamiento del lenguaje natural es según IBM “la rama de la informática -más concretamente, de la inteligencia artificial- que se ocupa de dotar a los ordenadores de la capacidad de entender el lenguaje hablado y escrito del mismo modo que los seres humanos. Esta tecnología ha alcanzado un alto nivel en la actualidad gracias a la aplicación de tecnologías como el *machine learning* (aprendizaje automático), el *big data*, el internet de las cosas o redes neuronales.”

La tecnología arriba mencionada tiene muchas aplicaciones actualmente, entre ellas podemos encontrar:

- Búsqueda avanzada de información: Permite detectar y recuperar automáticamente información relevante de documentos.

- *Chatbots*: Se trata del primer paso para desarrollar asistentes de voz.
- Clasificación automática de textos: Puede etiquetar textos automáticamente según su temática.
- Análisis de sentimientos: Por las palabras que se utilizan es capaz de evaluar las emociones u opiniones, y poder clasificarlas como positivo, negativo o neutral.

En nuestro caso, haremos uso del análisis de sentimientos para intentar predecir si la relación entre los principios activos será negativa, positiva o neutra.

En cuanto a fuentes o estudios similares en los que poder orientarnos, no se han encontrado demasiados. Sin embargo, podemos destacar los siguientes:

- *“INDI: a computational framework for inferring drug interactions and their associated recommendations”*

En este artículo se hace una predicción utilizando la farmacocinética y farmacodinámica, basadas en las propiedades como su estructura química, permeabilidad, solubilidad y polaridad de los fármacos relacionándolos con las enzimas principales (CYP) implicadas en el metabolismo del fármaco. Su método está basado en la similitud de unos y otros fármacos usando las propiedades que hemos comentado anteriormente. Además, de predecir su objetivo, proporciona información adicional como recomendaciones sobre qué hacer si se administran fármacos que se predijo que interaccionarían.

- *“Label Propagation Prediction of Drug-Drug Interactions Based on Clinical Side Effects”*

Las características utilizadas para la predicción son basadas en la similitud de los efectos adversos de los medicamentos y sus estructuras químicas.

- *“Analysis of drug combinations: current methodological landscape”*

Este artículo se basa en la predicción utilizando la relación de los efectos provocados por la combinación de los fármacos y las curvas dosis-efecto.

- *“Isobologram Analysis: A Comprehensive Review of Methodology and Current Research”*

Este estudio se apoya en el análisis de isobogramas y su aplicación en la interacción farmacológica, además de la similitud en el coeficiente de cototoxicidad y el índice de concentración inhibitoria fraccional.

Actualmente, existen distintas webs gratuitas donde consultar las interacciones entre principios activos pero, aunque son de acceso público, siempre recomiendan consultar con nuestro médico, ya que no se debe hacer uso por cuenta propia.

Finalmente, destacamos *DrugBank* y *Drugs.com*. como las fuentes más interesantes y de donde posteriormente, usaremos información sobre los principios activos.

Cómo podemos observar, todos los estudios realizados están enfocados desde una perspectiva más bien química y farmacológica, usando la estructura química de cada principio o sus gráficos dosis-efecto. En nuestro caso, nos enfocaremos en características relacionadas con la similitud de los principios desde el punto de vista de la semántica de las palabras, usando modelos de análisis de sentimientos.

Metodología

En cuanto a la metodología utilizada para el desarrollo del trabajo de fin de grado, hemos utilizado una metodología ágil mediante la cual con reuniones periódicas con la tutora se han definido los diferentes objetivos o tareas que se debían llevar a cabo y sus tiempos.

Para recabar y desarrollar el trabajo hemos hecho uso de recursos bibliográficos, recursos web y tutoriales sobre el uso y aplicación de técnicas del lenguaje de procesamiento natural.

Desarrollo de la propuesta

Diseño de la base de datos

Tras recopilar información sobre que son las interacciones y cómo se originan, pasamos a diseñar cómo sería la base de datos donde almacenaríamos la información.

Puesto que sabemos que los medicamentos están formados por principios activos responsables del origen de las interacciones, diseñamos la base de datos mediante un modelo relacional y un modelo entidad-relacional, que luego implementaremos en un *software* para poder manejar los datos.

Modelo entidad-relación:

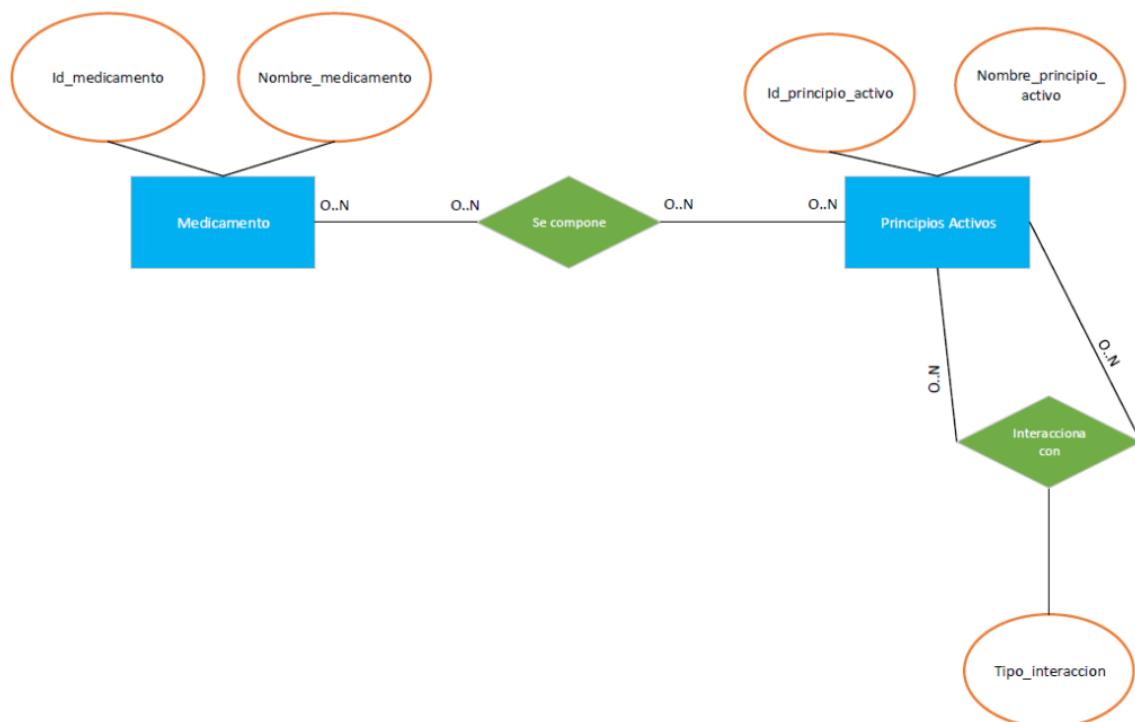


Ilustración 2. Modelo entidad-relación. Fuente: Elaboración propia

Modelo Relacional:

MEDICAMENTO

Id_medicamento -> CP

Nombre_medicamento

PRINCIPIO ACTIVO

Id_principio_activo -> CP

nombre_principio_activo

COMPOSICION

Id_medicamento, Id_principio_activo -> CP

Id_medicamento, Id_principio_activo -> CAj

INTERACCION

Id_principio_activo_1, Id_principio_activo_2 -> CP

Id_principio_activo_1, Id_principio_activo_2 -> CAj

Tras el diseño de la base de datos, pasaremos a implementarlo para poder gestionar y manipular la información. El *software* que usaremos será el gestor de base de datos HeidiSQL. HeidiSQL es según Ansgar Becker “un *software* libre y tiene como objetivo que sea fácil de aprender. "Heidi" le permite ver y editar datos y estructuras desde computadoras que ejecutan uno de los sistemas de base de datos MariaDB, MySQL, Microsoft SQL, PostgreSQL y SQLite. Inventado en 2002 por Ansgar, HeidiSQL pertenece a las herramientas más populares para MariaDB y MySQL en todo el mundo”.

Además, para su funcionamiento debemos instalar XAMPP, según Wikipedia “es un paquete de *software* libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl”. Sin él, no podemos poner a funcionar HeidiSQL.

Adjunto imágenes de cómo se crea una tabla, ya que el proceso para el resto de las tablas que forman la base de datos es el mismo:

- Agregamos los atributos de la tabla y el tipo de datos.



Columnas: + Agregar × Borrar ▲ Subir ▼ Bajar				
	#	Nombre	Tipo de datos	Longitud/Conjunto
	1	id_principio_activo_1	INT	11
	2	id_principio_activo_2	INT	11
	3	tipo_interaccion	VARCHAR	50

Ilustración 3. Atributos de la tabla. Fuente: Elaboración propia

- Añadimos las claves primarias y ajenas de la tabla.

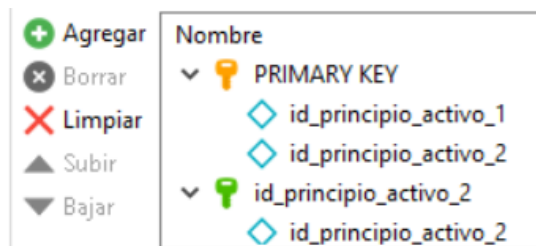


Ilustración 4. Añadir claves primarias. Fuente: Elaboración propia

- Referenciamos las tablas para poder relacionar unas con otras.

Nombre de la llave	Columnas	Tabla de refere...	Columnas ...	En UPDATE	En DELETE
id_principio_activo_1	id_principio_activo_1	medicamento...	id_principi...	RESTRICT	RESTRICT
id_principio_activo_2	id_principio_activo_2	medicamento...	id_principi...	RESTRICT	RESTRICT

Ilustración 5. Referenciar claves ajenas. Fuente: Elaboración propia

- Base de datos creada con todas las tablas

	medicamentos	71,9 MiB
	composicion	320,0 KiB
	interaccion	71,1 MiB
	medicamen...	368,0 KiB
	principios	48,0 KiB

Ilustración 6. Tablas de la base de datos. Fuente: Elaboración propia

Añadir información a la base de datos

Una vez creada la base de datos en MySQL, debemos completarla con los datos necesarios para crear nuestro corpus con el que poder dotar de información a nuestro modelo de predicción. Con este fin, hicimos uso de diversos recursos web y bibliográficos.

Por un lado, realizamos una búsqueda para encontrar documentación relacionada con interacciones ya conocidas, con el propósito de tener un punto de partida claro. Durante la investigación, nos dimos cuenta de que gran parte de la información disponible hacía referencia a las interacciones de tipo negativas, puesto que son las que conviene evitar. En cuanto a las interacciones neutras solamente encontramos un artículo, y respecto a interacciones positivas no obtuvimos resultados.

La documentación que nos situaba en la base de partida y con la cual rellenamos la base de datos fue la siguiente:

- Interacciones negativas: Hemos hecho uso de un libro publicado por la sociedad española de farmacología (SEFH), en concreto, Girona Brumós, L. (2020). *Interacciones Farmacológicas*. (Segunda Edición) Donde se detallan de cada principio activo las interacciones que se conocen, además de una descripción del riesgo que puede producir, recomendaciones a la hora de combinar los principios activos y observaciones sobre efectos adversos que han ocurrido.
- Interacciones neutras: Encontramos el artículo *Compatibilidad entre fármacos por vía intravenosa* (2005), donde se administraron una serie de combinaciones de fármacos a la vez y no causaban ninguna reacción favorable ni desfavorable en el paciente.

Al ser este último artículo posterior al libro, pudimos comprobar que algunas de las interacciones que se daban a conocer como neutras, con el tiempo se ha demostrado que son negativas.

Por último, con respecto a las interacciones positivas no se encontró información alguna. De hecho, la tutora Paloma Moreda preguntó a compañeros especializados respecto a bibliografía sobre este tipo de interacciones, pero no obtuvimos ninguna. Por ello, creamos nuestro propio corpus de interacciones positivas basándonos en los principios activos que no tienen una relación negativa ni neutra, y de este modo podríamos crear muestras positivas si los relacionáramos. Para una mejor comprensión agregamos un ejemplo:

PA1-PA2 tiene una relación negativa

PA1-PA3 tiene una relación neutra

PA2-PA4 tiene una relación negativa

Por tanto, podemos decir que la relación de PA3-PA4 será positiva, puesto que no tienen relación negativa ni neutra.

Con muestras de los tres tipos de interacciones, añadimos los datos de los nombres de principios activos, medicamentos que contienen esos principios activos, la composición de esos medicamentos y las parejas de principios que forman las interacciones que hemos obtenido.

Para ello, hicimos uso de los servicios web de CIMA. En concreto, de la tecnología REST API, para descargar la información de los medicamentos y poder añadirla a nuestra base de datos. Programamos diversas funciones con las que poder hacer uso del REST API, conectar con la base de datos y añadir la información necesaria.

El código que les mostramos a continuación, realiza la función de añadir automáticamente a la base de datos el nombre de los medicamentos asociados a un principio activo, el nombre del principio activo por el que está formado el medicamento y la composición del medicamento, es decir, principios activos por los que está formado el medicamento. Se puede encontrar el código en **Anexo 1**:

```
'''Funcion mediante la cual añadimos automaticamente a la base de datos el nombre de los medicamentos obtenidos, los principios activos y la composicion de los medicamentos'''
def insertarMedicamentos(x,y):
    conexion1=mysql.connector.connect(host="localhost",user="root",passwd="",database="medicamentos")#Nos conectamos a nuestra base de datos
    cursor1=conexion1.cursor()
    params = {"practivo":x,"pagina":y,"autorizados":"1","comerc":"1"} #Parametros que se le agregan a la url con el principio activo y solo medicamentos autorizados.
    r = requests.get('https://cima.aemps.es/cima/rest/medicamentos',params) #Obtenemos la pagina deseada y almacenamos el json en la variable r.
    r1 = json.loads(r.content) #Leemos el json y lo convertimos en un valor Python en este caso diccionarios.
    print("Pagina:",r1["pagina"])
    if r1["totalFilas"] != 0:
        print("Total Filas:",r1["totalFilas"])
        for z in r1["resultados"]:
            print("Medicamento:",z["nombre"])
            sql="insert into medicamento(id_medicamento,nombre_medicamento)values(%s,%s)"#Sentencia sql para añadir nuevos medicamentos
            filas=("",z["nombre"])#Tupla con todos los nombres de los medicamentos
            cursor1.execute(sql,filas)#Añadimos los nuevos medicamentos a la base de datos
            conexion1.commit()
            sql2=cursor1.lastrowid

            for x in z["vtm"][:]["nombre"].capitalize().split(sep='+ '):
                print("Principio activo:",x)

                sql="insert ignore into principios(id_principio_activo,nombre_principio_activo)values(%s,%s)"
                filas=("",x)
                cursor1.execute(sql,filas)
                conexion1.commit()

                sql="select id_principio_activo from principios where nombre_principio_activo = %s"
                cursor1.execute(sql,(x,))
                record = cursor1.fetchone()
                for y in record:
                    y

                sql="insert ignore into composicion(id_medicamento,id_principio_activo)values(%s,%s)"
                filas=(sql2,y)
                cursor1.execute(sql,filas)
                conexion1.commit()

    else:
        print("Ya no hay mas resultados")
```

Ilustración 7. Insertar medicamentos. Fuente: Elaboración propia

Adjuntamos una captura de cómo quedan los datos añadidos a la base de datos:

id_medicamento	nombre_medicamento
1	ABACAVIR/LAMIVUDINA DR. REDDYS 600 MG/3...
2	ABACAVIR/LAMIVUDINA KERN PHARMA 600 MG...
3	ABACAVIR/LAMIVUDINA SANDOZ 600 MG/300 ...

Ilustración 8. Medicamentos añadidos a la base de datos. Fuente: Elaboración propia

id_principio_activo	nombre_principio_activo
1	Abacavir
2	Abatacept
3	Abciximab

Ilustración 9. Principios activos añadidos a la base de datos. Fuente: Elaboración propia



id_medamento 	id_principio_activo 
1	1
1	411

Ilustración 10. Composición medicamentos añadidos a la base datos. Fuente: Elaboración propia

Seguramente muchos principios activos no estarán de inicio en la base de datos, ya que nos basamos en los nombres que aparecían en los documentos anteriormente mencionados y son de los que sabemos que existe interacciones entre sí. A medida que se añadan nuevos principios, utilizaremos nuestro modelo para predecir con que otros principios interaccionarán e ir aumentando nuestras muestras.

Para introducir las interacciones, hicimos uso de otro código en el que introducíamos los nombres de los dos principios y el tipo de interacción que presentan. Se puede encontrar el código en **Anexo 2**:

```
import mysql.connector

def insertarInteraccion(x,y,z):
    conexion1=mysql.connector.connect(host="localhost",user="root",passwd="",database="medicamentos")
    cursor1=conexion1.cursor()
    sql1="select id_principio_activo from principios where nombre_principio_activo = %s"
    cursor1.execute(sql1, (x,))
    record = cursor1.fetchone()
    print(record)
    for o in record:
        o

    sql2="select id_principio_activo from principios where nombre_principio_activo = %s"
    cursor1.execute(sql1, (y,))
    record2 = cursor1.fetchone()
    print(record2)
    for h in record2:
        h

    sql="insert into interaccion(id_principio_activo_1,id_principio_activo_2,tipo_interaccion)values(%s,%s,%s)"
    filas=(o,h,z)
    cursor1.execute(sql,filas)
    conexion1.commit()
```

Ilustración 11. Insertar interacciones. Fuente: Elaboración propia

Las interacciones de tipo positivas las introducimos mediante la función de importar archivos CSV de HeidiSQL y puesto que fue un archivo propio, no tuvimos que hacer uso de la función mostrada anteriormente.

Adjuntamos una captura de cómo queda la tabla una vez los datos son añadidos:

id_principio_activo_1	id_principio_activo_2	tipo_interaccion
1	1	Negativa
1	2	Positiva
1	3	Positiva

Ilustración 12. Interacciones añadidas a la base de datos. Fuente: Elaboración propia

Actualización de la base de datos

Una vez añadida la primera información a la base de datos, nos dimos cuenta de que la web CIMA se actualiza periódicamente con nuevos medicamentos. Por tanto, para tener la base de datos lo más actualizada posible, decidimos crear una función para añadir automáticamente los nuevos medicamentos, su composición, y en el caso de que el principio activo que forma a los nuevos medicamentos no esté añadido, lo añadía.

Función para actualizar la base de datos, se puede encontrar el código en **Anexo 3**:

```
'''Funcion medicamento la cual obtenemos una lista de los cambios introducidos en la base de datos de medicamentos de CIMA'''
def actualizacion(x,y):
    conexion=mysql.connector.connect(host="localhost",user="root",passwd="",database="medicamentos")
    cursor=conexion.cursor()
    params = {"fecha":x,"pagina":y}#Parametros que se le añaden a la url, en este caso Fecha a partir de la cual se desea conocer que medicamentos se han dado de alta
    r = requests.get('https://cima.aemps.es/cima/rest/registroCambios',params)
    r1 = json.loads(r.content)
    print("Pagina: ",r1["pagina"])#Mostramos el numero de pagina al que hemos accedido
    if r1["totalFilas"] != 0: #Indicamos que mientras el numero de filas sea distinto de zero aún quedan paginas por revisar
        print("Total Filas:",r1["totalFilas"])
        for y in r1["resultados"]:
            params = {"nregistro":y["nregistro"]}
            r = requests.get('https://cima.aemps.es/cima/rest/medicamentos',params)
            r1 = json.loads(r.content)
            for z in r1["resultados"]:
                if y["tipocambio"] == 1 and z["vtm"]["nombre"]!="null": #Si se han añadido nuevos medicamentos se mostraran por pantalla y serán añadidos a la base de datos
                    print("Medicamento:",z["nombre"])

                    sql="insert ignore into medicamento(id_medicamento,nombre_medicamento)values(%,%)"
                    filas=("",z["nombre"])
                    cursor.execute(sql,filas)
                    conexion.commit()
                    sql2=cursor.lastrowid

                    for x in z["vtm"]["nombre"].capitalize().split(sep=' '): #Si el principio activo que forma a los nuevos medicamentos no lo tenemos registrado serán añadido
                        print("Principio activo:",x)

                        sql="insert ignore into principios(id_principio_activo,nombre_principio_activo)values(%,%)"
                        filas=("",x)
                        cursor.execute(sql,filas)
                        conexion.commit()

                        sql="select id_principio_activo from principios where nombre_principio_activo = %s"
                        cursor.execute(sql,(x,))
                        record = cursor.fetchone()
                        for y in record:
                            y

                        sql="insert into composicion(id_medicamento,id_principio_activo)values(%,%)"
                        filas=(sql2,y)
                        cursor.execute(sql,filas)
                        conexion.commit()

    else:
        print("Ya no hay mas resultados")
```

Ilustración 13. Actualizar base de datos. Fuente: Elaboración propia

Desarrollo del modelo

Con la información en la base de datos de las interacciones que conocemos, vamos a pasar a desarrollar el modelo mediante el cual si se añade un principio activo nuevo que no tenemos añadido, predeciremos la relación que tendría con los otros principios.

Lo desarrollaremos haciendo uso de las técnicas de PNL. Como dijimos anteriormente en el procesamiento del lenguaje natural, es necesario tratar los datos de una forma específica para que un ordenador pueda interpretarlos de manera correcta. Utilizaremos NLTK, Scikit-learn y Pandas, tres de las librerías más populares en la ciencia de datos para el lenguaje Python.

- NLTK: Según Wikipedia “es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (PLN) simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra. Se acompaña de un libro que explica los conceptos subyacentes a las tareas de procesamiento del lenguaje compatibles el toolkit, además de programas de ejemplo. NLTK está destinado a apoyar la investigación y la enseñanza en procesamiento de lenguaje natural (PLN) o áreas muy relacionadas, que incluyen la lingüística empírica, las ciencias cognitivas, la inteligencia artificial, la recuperación de información, y el aprendizaje de la máquina.”
- Scikit-learn: Según Wikipedia “es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python.1 Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, *Gradient boosting*, *K-means* y DBSCAN.”
- Pandas: Según Wikipedia “es una biblioteca de software escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales.”

Para crear un modelo hay que dotarlo de información para que sea capaz de aprender. A tal efecto, el sistema debe ser capaz de interpretar los datos, debemos seguir unos pasos antes de poder usar la información en nuestro modelo:

- Recopilar los datos que necesitemos para dotar de información al sistema.
- Preprocesar los datos.
- Extraer vectores de características de los datos.
- Elegir el modelo o los modelos que queramos usar y entrenarlo.

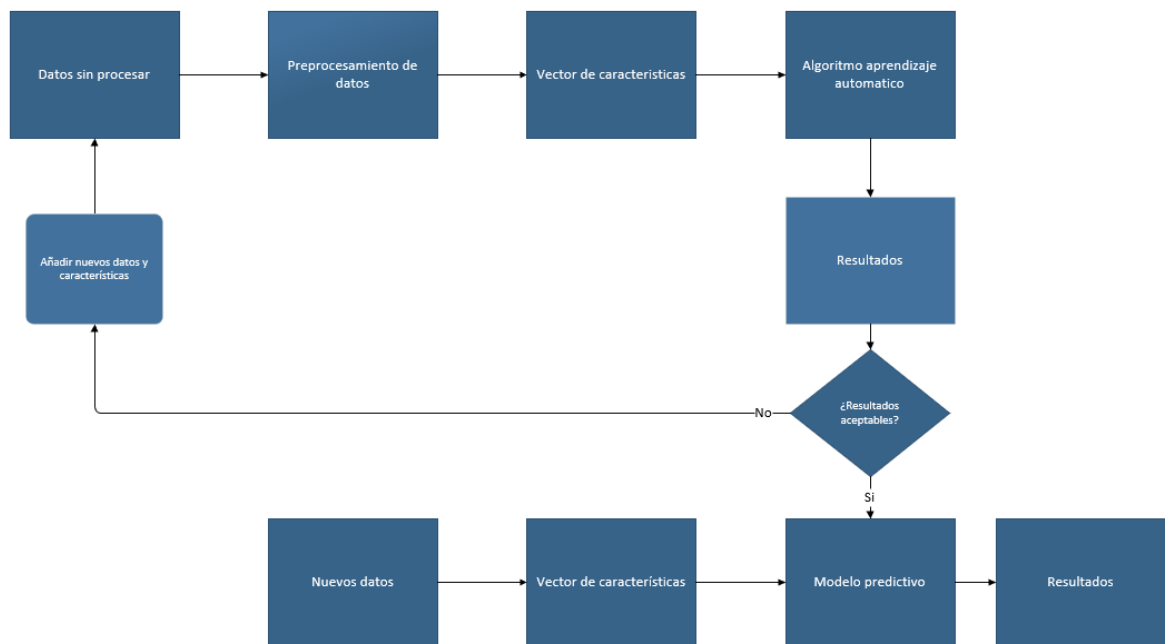


Ilustración 14. Diagrama modelo aprendizaje automático. Fuente: Elaboración propia

Durante el proceso, hemos realizado diversas pruebas hasta dar con una que ha conseguido dotar al modelo de suficiente información con la que aprender a distinguir si la relación entre los principios activos era positiva, negativa o neutra.

El primer modelo que creamos era muy simple, intentamos que el modelo con el de nombre los principios activos y las interacciones que teníamos fuera capaz de encontrar una similitud entre los pares de principios activos.

Explicaremos el código paso a paso para que se entienda cómo se fue desarrollando el modelo. Podemos encontrar el código completo en [Anexo 4](#):

Las librerías utilizadas fueron las siguientes:

```

import nltk
import pandas as pd
import csv
from nltk.stem import SnowballStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from csv import reader
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

```

Ilustración 15. Librerías. Fuente: Elaboración propia

La datos utilizados han sido las interacciones, para ello hemos extraído la información de la base de datos y los hemos exportado como formato CSV, con el fin de tratarlos con la librería pandas:

Metoclopram	Aciclovir	Neutro
Metoprolol	Aciclovir	Positiva
Metotrexato	Aciclovir	Negativa

Ilustración 16. Archivo CSV interacciones. Fuente: Elaboración propia

Con los datos necesarios en su correspondiente archivo, es esencial preprocesar los datos, Esto consiste en eliminar del texto todo aquello que no aporta información, por ejemplo, eliminar signos de puntuación y palabras que no aportan significado, entre otros. Junto a esto, se lleva a cabo la tokenización del texto, en el caso que tuvieramos una porción de texto, tokenizar consiste en dividir el texto en las unides que lo forman, en este caso, las unidades serían las palabras. Esta práctica es eficiente y conveniente ya que los ordenadores analizan los datos examinando qué palabras aparecen con más frecuencia.

Solo nos hizo falta convertir las mayúsculas en minúsculas, este procedimiento se sigue porque el sistema puede entender que una letra en mayúscula tiene más importancia y podría ser un problema. Lo ideal es dejar todo con el mismo formato, y nuestras palabras ya estaban tokenizadas.

Decidimos probar a derivar las palabras a su forma raíz, ya que algunos de los nombres de los principios activos son derivados unos de otros, y eso podría hacer que tuviera una similitud con la que el modelo encontrara un patrón por el que guiarse. Para ello, utilizamos “SnowballStemmer”.

```
stemmer = SnowballStemmer("spanish", ignore_stopwords=False)

data_1 = pd.read_csv('principios.csv')
data_1['nombre_principio_activo'] = data_1['nombre_principio_activo'].str.lower()

data = pd.read_csv('interaccion_nombre_prueba.csv')
data['nombre_principio_1'] = data['nombre_principio_1'].str.lower()
data['nombre_principio_1'] = data['nombre_principio_1'].apply(lambda x : stemmer.stem(x))
data['nombre_principio_2'] = data['nombre_principio_2'].str.lower()
data['nombre_principio_2'] = data['nombre_principio_2'].apply(lambda x : stemmer.stem(x))
data['tipo_interaccion'] = data['tipo_interaccion'].str.lower()
```

Ilustración 17. Código limpiar texto y leer datos. Fuente: Elaboración propia

nombre_principio_1	nombre_principio_2	tipo_interaccion
abacav	abacav	negativa
abatacept	abacav	positiva
abciximab	abacav	positiva
abirateron	abacav	positiva
acarab	abacav	positiva
...
clonixinato de lisin	fitomenadiador	positiva
clindamicin	fitomenadiador	positiva
fitomenadiador	fitomenadiador	positiva
sulfametoaxol	fitomenadiador	positiva
fitomenadiador	fitomenadiador	negativa

Ilustración 18. Datos limpios. Fuente: Elaboración propia

Una vez tenemos los datos limpios, debemos dividir nuestro dataset en dos bloques. Uno para entrenar el modelo y otro para testarlo mediante el `train_test_split` de `sklearn`. Después, pasaremos los datos a un formato que pueda entender un ordenador, ya que como sabemos, los ordenadores no entienden las palabras. Por tanto, las convertiremos en su representación numérica. Una forma sencilla es mediante el algoritmo “Term Frequency Inverse Document Frequency Vectorizer” (TF-IDF Vectorizer).

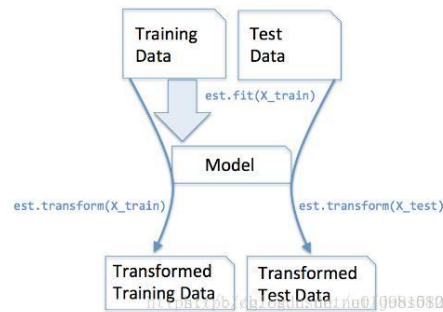


Ilustración 19. Funcionamiento TF-IDF.[Imagen]. En El proceso de cálculo de TF-IDF en `TfidfVectorizer` en `sklearn`. Fuente: <https://programmerclick.com/article/70301137743/>

TF-IDF transforma las palabras de una manera simple. Hemos de pasarle un vocabulario o corpus que son las palabras que más importancia deben tener, atribuye un valor a las palabras por la frecuencia que aparecen y crea una matriz dispersa que serán los valores que usaremos para entrenar a nuestro modelo.

```

train, test = train_test_split(data, test_size=0.20, random_state=0)

train_x_1, train_y = train['nombre_principio_1'], train['tipo_interaccion']
test_x_1, test_y = test['nombre_principio_1'], test['tipo_interaccion']

train_x_2, train_y = train['nombre_principio_2'], train['tipo_interaccion']
test_x_2, test_y = test['nombre_principio_2'], test['tipo_interaccion']

tfidf = TfidfVectorizer(stop_words=stop_word, vocabulary = vocabulario) #Aquí se le proporciona el vocabulario
train_x_3 = tfidf.fit_transform(train_x_1)
test_x_3 = tfidf.transform(test_x_1)
train_x_4 = tfidf.transform(train_x_2)
test_x_4 = tfidf.transform(test_x_2)

train_x_vector = train_x_3.multiply(train_x_4)
test_x_vector = test_x_3.multiply(test_x_4)

```

Ilustración 20. TF-IDF configuración. Fuente: Elaboración propia

El siguiente paso es elegir el modelo. Escogimos los siguientes:

- *Support-vector machines* o SVM.
- Árboles de decisión o *Decision Tree*.
- Regresión lineal.

Para la configuración de los modelos le aplicamos el parámetro “balanced”. Debido a que la diferencia de muestras entre positivas, negativas y neutras es demasiado grande (tenemos 132756

positivas, 7979 negativas y 71 neutras) , esto puede ser un problema a la hora de que clasificar. La función del parámetro es equilibrar el peso de cada tipo de muestra y ajustarlos automáticamente.

```
"""Suport Vector Machine (SVM)"""
svc = SVC(kernel='linear', class_weight="balanced")
svc.fit(train_x_vector, train_y)

"""Decision Tree"""
dec_tree = DecisionTreeClassifier(class_weight="balanced")
dec_tree.fit(train_x_vector, train_y)

"""Logistic Regression"""
lr = LogisticRegression(class_weight="balanced")
lr.fit(train_x_vector, train_y)
```

Ilustración 21. Modelos de predicción. Fuente: Elaboración propia

Tras ejecutar los modelos, obtuvimos las siguientes métricas con los resultados :

- Acuraccy
- F1 Score
- Reporte Clasificación
- Confusion Matrix

```
Score (Acuraccy)
SVC:  0.0005042398761416417
DEC:  0.0005042398761416417
LR:   0.0005042398761416417
```

Ilustración 22. Score primera prueba. Fuente: Elaboración propia

```
F1 Score
SVC:  [0.      0.      0.00100797]
DEC:  [0.      0.      0.00100797]
LR:   [0.      0.      0.00100797]
```

Ilustración 23. F1 Score primera prueba. Fuente: Elaboración propia

SVC:	precision	recall	f1-score	support
positiva	0.00	0.00	0.00	132756
negativa	0.00	0.00	0.00	7979
neutro	0.00	1.00	0.00	71
accuracy			0.00	140806
macro avg	0.00	0.33	0.00	140806
weighted avg	0.00	0.00	0.00	140806
DEC:	precision	recall	f1-score	support
positiva	0.00	0.00	0.00	132756
negativa	0.00	0.00	0.00	7979
neutro	0.00	1.00	0.00	71
accuracy			0.00	140806
macro avg	0.00	0.33	0.00	140806
weighted avg	0.00	0.00	0.00	140806
LR:	precision	recall	f1-score	support
positiva	0.00	0.00	0.00	132756
negativa	0.00	0.00	0.00	7979
neutro	0.00	1.00	0.00	71
accuracy			0.00	140806
macro avg	0.00	0.33	0.00	140806
weighted avg	0.00	0.00	0.00	140806

Ilustración 24. Reporte clasificación primera prueba. Fuente: Elaboración propia

```

Confusion Matrix
SVC: [[ 0 0 132756]
      [ 0 0 7979]
      [ 0 0 71]]
DEC: [[ 0 0 132756]
      [ 0 0 7979]
      [ 0 0 71]]
LR:  [[ 0 0 132756]
      [ 0 0 7979]
      [ 0 0 71]]

```

Ilustración 25. Confusion matrix primera prueba. Fuente: Elaboración propia

Como podemos observar, los resultados son desfavorables. Ninguno de los tres modelos ha conseguido aprender a clasificar. Por tanto, debemos proporcionarle más información al modelo para que mejore. Para que sean resultados aceptables, la diagonal de la matriz debe tener el mayor número posible, puesto que hace referencia a las interacciones de cada tipo que clasifica correctamente.

Investigando características que podríamos añadir a nuestro dataset para mejorar el modelo, nos dimos cuenta de que los principios activos que pertenecen al mismo grupo farmacológico Ilustración 26, tienen la misma terminación la mayoría de ellos.

Prefijo-	Grupo farmacológico (acciones)	-Sufijo
	Antiinflamatorios del grupo ibufenac, derivado del ácido fenilacético	-ac
	Analgésicos antiinflamatorios del grupo de la fenilbutazona	-butazona
Fenamato	Antiinflamatorios derivados del ácido antranílico, como el meclofenamato	Fenamato
	Antiinflamatorios del grupo de la indometacina	-metacina
Sal-	Derivados del ácido salicílico (antiinflamatorio, antipirético, analgésico)	
	Antiinflamatorios del grupo ibuprofeno, del ácido arilpropiónico	-profeno
	Analgésicos del grupo de meperidina	-eridina
	Analgésico (agonistas opioides), como el tapentadol o tramadol	-adol
	Prostaglandinas, como el latanoprost (anti-glaucoma)	-prost
	Anestésicos locales.	-caina
	Ansiolíticos de estructura no benzodiazepínica	-azolam, -azam
	Sustancias del grupo diazepam de las benzodiazepinas	-acepam
	Antagonistas/ agonistas del grupo de los benzomorfanos (opioides)	-azocina
	Ansiolíticos derivados del propanodiol y pentanodiol	-bamato
Barb-	Derivados del ácido barbitúrico (hipnóticos).	
Nal-	Agonistas/antagonistas narcóticos del grupo de las normorfinas	
Orfan-	Agonistas/antagonistas narcóticos del grupo morfina	
	Neurolépticos del grupo butirofenonas	-perona, -peridol
	Derivados de sulpirida (neurolépticos)	-prida
Andr-	Esteroides andrógenos	
Gest-	Esteroides progestágenos	
Estr-	Estrógenos	
	Esteroides andrógenos, como abiteratona	-terona
	Análogos de somatostatina, como octreótide, lanreótide y	-ótide
	Esteroides de uso tópico del grupo acetal, como fluocinolona y triamcinolona acetónido	-ónido
Cef-	Antibióticos derivados del ácido cefalosporánico	
	Antibióticos derivados del grupo de la tetraciclina	-ciclina
	Antibióticos naturales que no pertenecen a una clase determinada	-cidina
	Antibióticos derivados del ácido 6-aminopenicilánico, como la penicilina	-cillina
	Antibióticos producidos por Streptomyces	-micina
	Antibacterianos derivados del ácido nalidixico	-oxacina
	Derivados de 4,4-diaminodifenilsulfona (antibiótico, bacteriostático, antituberculoso)	-dapsona
Nifur-	Derivados del 5-nitrofurano (antibiótico), como la nitrofuraxona o nitrofurantoina	
	Antiprotozoarios (anti-parásitos) pertenecientes al grupo metronidazol	-nidazol
	Derivados de la quinolina (quinina) (antipalúdica)	-quina
Vir-	Sustancias antivirales	
	Antihelmínticos (anti-gusanos) que no sean de un grupo determinado	-antel
	Antihelmínticos (anti-gusanos) del grupo mebendazol	-bendazol
	Derivados de la acridina (antineoplásico)	-crina
Sulfa-	Quimioterapia con sulfonamidas	
Mito-	Antineoplásicos nucleotóxicos	
	Antagonista del calcio del grupo de la nifedipina	-dipina
	Simpaticomiméticos adrenérgicos	-drina
	Antiaritmico/antianginoso/antihipertensivos, como la hidralazina	-lazina
	Agentes bloqueantes Beta adrenérgicos del grupo propranolol	-olol
	Broncodilatadores del grupo de la fenetilamina	-prenalina
	Vasoconstrictores	-presina
	Antihipertensores del grupo captopril	-pril
	Broncodilatadores derivados de la fetilamina, como albuterol	-terol
	Anoréxicos (provocan saciedad, para no comer) del grupo de la fenetilamina	-orex
	Antihistamínicos H ₂ derivados de la cimetidina (H ₂ gástricos)	-tinidina
Stat	Inhibidores enzimáticos	stat

Tabla PRO1.2. AFIJOS útiles en MEDICAMENTOS. Fuente: United States Adopted Names Council (USAN). Disponible en: <http://ow.ly/xark3080YdJ>.

Ilustración 26. Prefijos y sufijos útiles en medicamentos. Core Radiológico. (2011). Prefijos y sufijos útiles en medicamentos [Fotografía]. URL: <http://www.coreradiologico.com/contenidos/zonas/detalleApartado.aspx?idAPadre=xx0FI7%2FcbVfwa0MvdU%2FWSA%3>

Esto puede ser interesante porque los principios activos que sean del mismo grupo deben tener interacciones de los distintos tipos en común. Así que añadimos esta información para comprobar si los resultados mejoran.

Los cambios con respecto al código anterior son los siguientes. Podemos encontrar el código completo en el **Anexo 5**:


```

data_1 = pd.read_csv('principios_terminaciones_1.csv')
vocabulario = data_1['terminaciones'].unique().tolist()

data = pd.read_csv('interaccion_terminaciones.csv', sep=';')
data['nombre_principio_1'] = data['nombre_principio_1'].str.lower()
data['terminacion_1'] = data['terminacion_1'].str.lower()
data['nombre_principio_2'] = data['nombre_principio_2'].str.lower()
data['terminacion_2'] = data['terminacion_2'].str.lower()
data['tipo_interaccion'] = data['tipo_interaccion'].str.lower()

data['Concat_1'] = data['nombre_principio_1'] + " " + data['terminacion_1']
data['Concat_2'] = data['nombre_principio_2'] + " " + data['terminacion_2']

```

Ilustración 27. Introducir datos terminaciones y preprocesarlos. Fuente: Elaboración propia

Añadimos el nuevo dataset, se preprocesó el texto y cambiamos el vocabulario por el nuevo.

nombre_principio_1	terminacion_1	nombre_principio_2	terminacion_2	tipo_interaccion
Abacavir	avir	Abacavir	avir	Negativa
Abatacept	cept	Abacavir	avir	Positiva
Abciximab	ximab	Abacavir	avir	Positiva
Abiraterona	terona	Abacavir	avir	Positiva

Ilustración 28. Archivo CSV interacciones con terminaciones. Fuente: Elaboración propia

id_principio_1	nombre_principio_1	terminaciones
1	Abacavir	avir
2	Abatacept	cept
3	Abciximab	ximab
4	Abiraterona	terona
5	Acarbosa	osa
6	Aceclofenaco	aco

Ilustración 29. Archivo CSV principio activo-terminaciones. Fuente: Elaboración propia

Los resultados obtenidos de añadir las terminaciones al modelo son:

```

Score (Acuraccy)
SVC: 0.9374585729863799
DEC: 0.9376082447775236
LR: 0.937408682389332

```

Ilustración 30. Score segunda prueba. Fuente: Elaboración propia

```

F1 Score
SVC: [0.96768463 0.05493188 0.01769912]
DEC: [0.967765 0.05464481 0.01769912]
LR: [0.967658 0.05489 0.01769912]

```

Ilustración 31. F1 Score segunda prueba. Fuente: Elaboración propia

```

Reporte Clasificacion
SVC:
      precision    recall  f1-score   support

   positiva      0.94      0.99      0.97    132026
   negativa      0.26      0.03      0.05     8204
     neutro      0.03      0.01      0.02        77

   accuracy              0.94    140307
  macro avg      0.41      0.35      0.35    140307
 weighted avg      0.90      0.94      0.91    140307

DEC:
      precision    recall  f1-score   support

   positiva      0.94      0.99      0.97    132026
   negativa      0.26      0.03      0.05     8204
     neutro      0.03      0.01      0.02        77

   accuracy              0.94    140307
  macro avg      0.41      0.35      0.35    140307
 weighted avg      0.90      0.94      0.91    140307

LR:
      precision    recall  f1-score   support

   positiva      0.94      0.99      0.97    132026
   negativa      0.26      0.03      0.05     8204
     neutro      0.03      0.01      0.02        77

   accuracy              0.94    140307
  macro avg      0.41      0.35      0.35    140307
 weighted avg      0.90      0.94      0.91    140307

```

Ilustración 32. Reporte de clasificación segunda prueba. Fuente: Elaboración propia

```

Confusion Matrix
SVC: [[131279    719     28]
      [ 7945    252      7]
      [   76      0      1]]
DEC: [[131302    696     28]
      [ 7947    250      7]
      [   76      0      1]]
LR:  [[131272    726     28]
      [ 7945    252      7]
      [   76      0      1]]

```

Ilustración 33. Confusion matrix segunda prueba. Fuente: Elaboración propia

Se puede observar una mejoría con respecto al anterior, pero sigue siendo necesario aportar más información al modelo, pues los resultados no son aceptables para que podamos considerar que clasifica correctamente.

El modelo SVC logró la mejor clasificación con 131279 positivas, 252 negativas y 1 neutra.

Como con las terminaciones el modelo mejoró, decidimos recoger toda la información posible sobre los principios activos, ya que todos los principios activos del mismo grupo farmacológico deberían tener características similares.

Mediante una de las páginas que ya mencionamos anteriormente: DrugBank (que se trata de una base de datos online sobre medicamentos), pudimos obtener nuevas características. Las que nos resultaron interesantes y que podrían darnos mejor resultado fueron las siguientes:

- *Summary*: Resumen de la función del principio activo.
- *Background*: Breve descripción con un poco más de detalle que el *Summary*.
- *Indication*: Indicaciones de uso sobre el principio activo.
- *Pharmacodynamics*: Efectos farmacológicos del principio activo en el organismo.
- *Mechanism of action*: Cómo actúa el principio activo.

Proporcionamos una captura sobre cómo se encuentra esta información en la web:

Abacavir	
Summary	Abacavir is an antiviral nucleoside reverse transcriptase inhibitor used in combination with other antiretrovirals for the treatment of HIV.
Background	Abacavir (ABC) is a powerful nucleoside analog reverse transcriptase inhibitor (NRTI) used to treat HIV and AIDS. Chemically, it is a synthetic carbocyclic nucleoside and is the enantiomer with 1S, 4R absolute configuration on the cyclopentene ring. In vivo, abacavir sulfate dissociates to its free base, abacavir.
Indication	Abacavir is indicated in combination with other anti-retroviral agents for the treatment of HIV-1 infection. ⁴ It is available in a combination product alongside dolutegravir and lamivudine for the treatment of adult and pediatric patients with HIV-1 who weight ≥ 10 kg. ³
Pharmacodynamics	Abacavir is a nucleoside reverse transcriptase inhibitor (NRTI) with activity against Human Immunodeficiency Virus Type 1 (HIV-1). Abacavir is phosphorylated to active metabolites that compete for incorporation into viral DNA. They inhibit the HIV reverse transcriptase enzyme competitively and act as a chain terminator of DNA synthesis. The concentration of drug necessary to effect viral replication by 50 percent (EC50) ranged from 3.7 to 5.8 μM ($1 \mu\text{M} = 0.28 \text{ mcg/mL}$) and 0.07 to 1.0 μM against HIV-1IIB and HIV-1BaL, respectively, and was $0.26 \pm 0.18 \mu\text{M}$ against 8 clinical isolates. Abacavir had synergistic activity in cell culture in combination with the nucleoside reverse transcriptase inhibitor (NRTI) zidovudine, the non-nucleoside reverse transcriptase inhibitor (NNRTI) nevirapine, and the protease inhibitor (PI) amprenavir; and additive activity in combination with the NRTIs didanosine, emtricitabine, lamivudine, stavudine, tenofovir, and zalcitabine.
Mechanism of action	Abacavir is a carbocyclic synthetic nucleoside analogue and an antiviral agent. Intracellularly, abacavir is converted by cellular enzymes to the active metabolite carbovir triphosphate, an analogue of deoxyguanosine-5'-triphosphate (dGTP). Carbovir triphosphate inhibits the activity of HIV-1 reverse transcriptase (RT) both by competing with the natural substrate dGTP and by its incorporation into viral DNA. Viral DNA growth is terminated because the incorporated nucleotide lacks a 3'-OH group, which is needed to form the 5' to 3' phosphodiester linkage essential for DNA chain elongation.

Ilustración 34. Información sobre principios activos DrugBank. Fuente: <https://go.drugbank.com/drugs/DB01048>

Después de añadir toda la información sobre los principios activos, nos dispusimos a usarla en el modelo. Esta vez fue necesario un mayor preprocesado del texto, ya que tenía signos de puntuación, palabras innecesarias y símbolos. En este sentido, actualizamos el código con respecto a los anteriores. Se puede encontrar el código completo en **Anexo 6**:

```

stop_word = stopwords.words('english')

data = pd.read_csv('principios_caracteristicas_1.csv', sep=';')

data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()
data['Summary'] = data['Summary'].str.lower()
data['Summary'] = data['Summary'].str.replace('[^\w\s]','',regex=True)
data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
data['Background'] = data['Background'].str.lower()
data['Background'] = data['Background'].str.replace('[^\w\s]','',regex=True)
data['Background'] = data['Background'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
data['Indication'] = data['Indication'].str.lower()
data['Indication'] = data['Indication'].str.replace('[^\w\s]','',regex=True)
data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()
data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]','',regex=True)
data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()
data['Mechanism_of_action'] = data['Mechanism_of_action'].str.replace('[^\w\s]','',regex=True)
data['Mechanism_of_action'] = data['Mechanism_of_action'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
data['terminación'] = data['terminación'].str.lower()

```

Ilustración 35. Código actualizado preprocesado del texto. Fuente: Elaboración propia

Así quedaría el texto una vez se aplica el preprocesado:

```

>>> print(data['Concat_1'][0])
abacavir abacavir antiviral nucleoside reverse
r nrti used treat hiv aids chemically synthetic
ent hiv 1 infection combination antiretroviral
tive metabolites compete incorporation viral dn

```

Ilustración 36. Datos limpios tercera prueba. Fuente: Elaboración propia

Con el texto limpio, pasamos toda la información como vocabulario al TF-IDF y ejecutamos los modelos para obtener los siguientes resultados:

```

Score (Acuraccy)
SVC:  0.929511713599464
DEC:  0.9724960265703065
LR:   0.8018844391227807

```

Ilustración 37. Score tercera prueba. Fuente: Elaboración propia

```

F1 Score
SVC:  [0.96370168 0.08928764 0.03813038]
DEC:  [0.98536508 0.77387949 0.66666667]
LR:   [0.88807834 0.17970361 0.02663533]

```

Ilustración 38. F1 Score tercera prueba. Fuente: Elaboración propia

```

Reporte Clasificacion
SVC:
      precision    recall  f1-score   support

   positiva      0.94      0.98      0.96    132026
   negativa      0.39      0.05      0.09     8204
    neutro       0.02      0.40      0.04         77

   accuracy              0.93    140307
  macro avg      0.45      0.48      0.36    140307
 weighted avg      0.91      0.93      0.91    140307

DEC:
      precision    recall  f1-score   support

   positiva      0.99      0.98      0.99    132026
   negativa      0.75      0.79      0.77     8204
    neutro       0.67      0.66      0.67         77

   accuracy              0.97    140307
  macro avg      0.80      0.81      0.81    140307
 weighted avg      0.97      0.97      0.97    140307

LR:
      precision    recall  f1-score   support

   positiva      0.95      0.83      0.89    132026
   negativa      0.12      0.34      0.18     8204
    neutro       0.01      0.44      0.03         77

   accuracy              0.80    140307
  macro avg      0.36      0.54      0.36    140307
 weighted avg      0.90      0.80      0.85    140307

```

Ilustración 39. Reporte clasificación tercera prueba. Fuente: Elaboración propia

```

Confusion Matrix
SVC: [[129973    634   1419]
      [ 7692    413     99]
      [   46      0     31]]
DEC: [[129879   2123     24]
      [ 1685   6518      1]
      [   26      0     51]]
LR:  [[109687  20044   2295]
      [ 5268   2789    147]
      [   40      3     34]]

```

Ilustración 40. Confusion matrix tercera prueba. Fuente: Elaboración propia

Cómo se puede observar, hay una gran mejoría con respecto a los dos anteriores. El modelo de árboles de decisión es el que mejores resultados obtuvo, clasificando 129879 positivas, 6518 negativas y 51 neutras.

También comentar que la diferencia de muestras puede afectar a que el modelo se incline por clasificar por el tipo mayoritario, aún utilizando el parámetro “balanced”, pero para mejorar eso sería necesario encontrar nuevas muestras tanto de interacciones negativas y neutras, ya que la diferencia es muy grande.

Luego de entrenar el modelo, lo guardaremos usando la librería pickle y crearemos una nueva función para poder usarlo cuando nos haga falta.

Hemos de guardar tanto el resultado del TF-IDF y en nuestro caso, el modelo de árboles de decisión, dado que es el que mejor resultados nos dio. Esto se hace para no tener que volver a entrenar al modelo cada vez que se quiera hacer uso de él, debido a que entrenarlo ha llevado hasta 5 días de espera.

A continuación, vamos a mostrar las funciones y explicar cada una. Hemos intentado hacerlo lo más modular posible ya que si se tuviese que introducir alguna modificación no sería necesario realizar cambios en todo el código. Se puede encontrar en el [Anexo 7](#) y [Anexo 8](#):

1. Proporcionamos el nombre del nuevo principio activo que acabamos de obtener de la función actualización que se explicó anteriormente.

```
from production_utils_1 import load_model, get_drugname, get_description, get_clfdata, update_data

## preparar datos

# obtener datos a clasificar
newdrugname = get_drugname()

def get_drugname():
    """
    Consultar BD, API, etc.
    """
    return "Loperamida" |
```

Ilustración 41. Nuevo principio activo. Fuente: Elaboración propia

2. Actualizamos nuestro archivo CSV con las características del nuevo principio. Se tiene que añadir de manera manual, (está pensado para que se actualizara de manera automática pero no hemos obtenido acceso a la base de datos), aunque no se descarta una implementación a futuro.

```
# actualizar csv
update_data(newdrugname)

def update_data(drugname):
    """
    Obtiene la información relevante (Summary, Background, etc.)
    """

    path = 'principios_caracteristicas_1.csv'
    data = pd.read_csv(path, sep=';', encoding='cp1252')
    drugdata = { 'id_principio_activo': 15001,
                  'nombre_principio_activo': drugname,
                  'Summary': ' Loperamide is a long acting antidiarrheal used to
                  'Background': 'One of the long-acting synthetic antidiarrheals
                  'Indication': 'For the control and symptomatic relief of acute
                  'Pharmacodynamics': 'Loperamide is a synthetic anti-diarrheal
                  'Mechanism_of_action': "In vitro and animal studies show that
                  'terminacion': 'amida'
                }

    data = data.append(drugdata, ignore_index=True)
    data.to_csv(path, sep=';', encoding='cp1252')
```

Ilustración 42. Añadir características del nuevo principio. Fuente: Elaboración propia

3. Preparar los datos, es decir, hacer el preprocesamiento (limpieza, extraer características, etc.).


```
# preparar datos (extraer características, etc.)
newdrugdesc= get_description(newdrugname)

def get_description(drugname):
    """
    Devuelve características (de momento es una maqueta)
    """

    stop_word = stopwords.words('english')

    data = pd.read_csv('principios_caracteristicas_1.csv', sep=';', encoding='cp1252')
    data = data[data['nombre_principio_activo']==drugname] # debería ser una sola fila

    data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()

    data['Summary'] = data['Summary'].str.lower()
    data['Summary'] = data['Summary'].str.replace('[^\w\s]',' ', regex=True)
    data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Background'] = data['Background'].str.lower()
    data['Background'] = data['Background'].str.replace('[^\w\s]',' ', regex=True)
    data['Background'] = data['Background'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Indication'] = data['Indication'].str.lower()
    data['Indication'] = data['Indication'].str.replace('[^\w\s]',' ', regex=True)
    data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()
    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]',' ', regex=True)
    data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()
    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.replace('[^\w\s]',' ', regex=True)
    data['Mechanism_of_action'] = data['Mechanism_of_action'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['termination'] = data['termination'].str.lower()

    data['Concat'] = data['nombre_principio_activo'] + " " + data['Summary'] + " " + data['Background'] + " " + data['Indication'] + " " + data[

    return data['Concat'].values[0]
```

Ilustración 43. Preprocesar los nuevos datos. Fuente: Elaboración propia

- Una vez tenemos los datos limpios, prepararemos la tabla que usaremos para la clasificación. En ella, se encontrarán el principio nuevo con sus características frente a los que ya disponemos.

```
# preparar datos para clasificación
clfddata = get_clfddata(newdrugdesc, newdrugname)

def get_clfddata(drugfeatures, drugname):
    """
    Obtiene la tabla para clasificar
    """

    stop_word = stopwords.words('english')

    data = pd.read_csv('principios_caracteristicas_1.csv', sep=';', encoding='cp1252')

    data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()
    data['Summary'] = data['Summary'].str.lower()
    data['Summary'] = data['Summary'].str.replace('[^\w\s]',' ', regex=True)
    data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Background'] = data['Background'].str.lower()
    data['Background'] = data['Background'].str.replace('[^\w\s]',' ', regex=True)

    data['Background'] = data['Background'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Indication'] = data['Indication'].str.lower()
    data['Indication'] = data['Indication'].str.replace('[^\w\s]',' ', regex=True)

    data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()
    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]',' ', regex=True)
    data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))
    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()
    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.replace('[^\w\s]',' ', regex=True)
    data['Mechanism_of_action'] = data['Mechanism_of_action'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

    data['Comp2'] = data['nombre_principio_activo'] + " " + data['Summary'] + " " + data['Background'] + " " + data['Indication']
    data['Comp1'] = drugfeatures

    data['Comp1Name'] = drugname
    data['Comp2Name'] = data['nombre_principio_activo']

    return data.loc[:, ['Comp1Name', 'Comp2Name', 'Comp1', 'Comp2']]
```

Ilustración 44. Preparar datos para la clasificación. Fuente: Elaboración propia

Se vería de la siguiente forma:

```
0  loperamida loperamide long acting antidiarrhea... abacavir abacavir antiviral nucleoside reverse...
1  loperamida loperamide long acting antidiarrhea... abatacept abatacept disease modifying antirheu...
2  loperamida loperamide long acting antidiarrhea... abciximab abciximab monoclonal anti glycoprote...
3  loperamida loperamide long acting antidiarrhea... abiraterona abiraterone antiandrogen used trea...
4  loperamida loperamide long acting antidiarrhea... acarbose acarbose alpha glucosidase inhibitor ...
```

Ilustración 45. Tabla para la clasificación. Fuente: Elaboración propia

5. Cargamos los modelos:

```
def load_model(rutaModelo = "pickle_model.pkl"):
    """
    Lee modelos serializados
    """
    with open(rutaModelo, 'rb') as file:
        pickle_model = pickle.load(file)
        return pickle_model

# leer modelos
tfidfpath = 'tfidf_todo.pkl'
tfidf = load_model(tfidfpath)

clfpath = 'dec tree_todo.pkl'
clf = load_model(clfpath)
```

Ilustración 46. Cargar modelos. Fuente: Elaboración propia

6. Transformamos la tabla en vectores mediante TF-IDF:

```
# aplicar funciones de extracción de características

tfcomp1 = tfidf.transform(clfdata['Comp1'])
tfcomp2 = tfidf.transform(clfdata['Comp2'])
comp1Name = clfdata['Comp1Name']
comp2Name = clfdata['Comp2Name']

features = tfcomp1.multiply(tfcomp2)
```

Ilustración 47. Transformar a vector de características. Fuente: Elaboración propia

7. Predecimos la relación que tendrá el principio activo nuevo con los que ya teníamos:

```
# clasificar
pred = clf.predict(features)

for c1n, c2n, p in zip(comp1Name, comp2Name, pred):
    print(f'{c1n} - {c2n} -> {p}')
```

Ilustración 48. Predecir relaciones del nuevo principio. Fuente: Elaboración propia

Caso de uso

Seguidamente, vamos a poner el modelo en funcionamiento con un caso real y predeciremos qué tipo de relación tendrá el nuevo principio activo con los que disponemos en la base de datos.

En primer lugar, haremos uso de la función “actualización”. Hemos seleccionado la fecha más cercana que se introdujo un nuevo medicamento, fue el 27/04/2022.

```
>>> actualizacion("27/04/2022",3)
Pagina: 3
Total Filas: 802
Medicamento: TAVNEOS 10 MG CAPSULAS DURAS
Principio activo: Avacopán
```

Ilustración 49. Nuevo medicamento añadido a CIMA. Fuente: Elaboración propia

Se añade automáticamente a la base de datos:

4.011	TAVNEOS 10 MG CAPSULAS DURAS	866	Avacopán	4.011	866
-------	------------------------------	-----	----------	-------	-----

Ilustración 50. Medicamento añadido a la base de datos

Ahora consultando DrugBank, añadiremos las características del Avacopán a nuestro dataset:

```
path = 'principios_caracteristicas_1.csv'
data = pd.read_csv(path, sep=';', encoding='cp1252')
drugdata = { 'id_principio_activo': 866,
             'nombre_principio_activo': drugname,
             'Summary': ' Avacopan is an orally bioavailable complement 5a recep
             'Background': 'Anti-neutrophil cytoplasmic (auto)antibody (ANCA)-as
             'Indication': 'Avacopan is indicated for the adjunctive treatment c
             'Pharmacodynamics': 'Avacopan is a complement 5a receptor (C5aR) ar
             'Mechanism_of_action': "Anti-neutrophil cytoplasmic (auto)antibody
             'terminación': 'copan'
           }
data = data.append(drugdata, ignore_index=True)
data.to_csv(path, sep=';', encoding='cp1252')
```

Ilustración 51. Características del nuevo principio. Fuente: Elaboración propia

Nuestro CSV, que contiene las características de los principios, se ha actualizado.

866	Avacopan	Avacopan is a	Anti-neutroph	Avacopan is ir	Avacopan is a	Anti-neutroph	copan
-----	----------	---------------	---------------	----------------	---------------	---------------	-------

Ilustración 52. Dataset actualizado con nueva información. Fuente: Elaboración propia

A continuación, ejecutaremos la predicción del modelo. Esta sería una pequeña muestra, ya que la salida esperada es más de 850 líneas:

```
Avacopan - abacavir -> positiva
Avacopan - abatacept -> negativa
Avacopan - abciximab -> positiva
Avacopan - abiraterona -> positiva
Avacopan - acarbosa -> positiva
Avacopan - aceclofenaco -> positiva
Avacopan - aceite esencial de menta piperita -> positiva
Avacopan - acenocumarol -> positiva
Avacopan - acetilsalicílico ácido -> positiva
Avacopan - aciclovir -> positiva
Avacopan - acitretina -> negativa
Avacopan - interferones -> negativa
Avacopan - ipilimumab -> negativa
Avacopan - irbesartan -> positiva
Avacopan - irinotecan -> positiva
Avacopan - isavuconazol -> positiva
Avacopan - isoniazida -> positiva
Avacopan - isonixina -> positiva
Avacopan - isoprenalina -> negativa
```

Ilustración 53. Resultados de las nuevas interacciones. Fuente: Elaboración propia

Análisis de costes

Planificación del proyecto

El proyecto de fin de grado se dividió en varias fases.

1. Recopilación de información.

Fue la primera fase y consistió en una búsqueda de información sobre qué eran las interacciones entre medicamentos, los efectos adversos y estudios relacionados sobre la predicción de interacciones.

Esta fase ha tenido lugar a lo largo de todo el trabajo por necesidad de encontrar nueva información.

Duración aproximada fue de 130 h.

2. Diseño y automatización de la base de datos.

Teniendo los conocimientos proporcionados en el grado sobre el diseño de base de datos, esta tarea fue, en parte, sencilla de realizar. En cuanto a encontrar la información que añadiríamos y cómo introducirla de manera automática, si fue necesario hacer uso de tutoriales y recursos webs sobre cómo usar un API en Python. Además de recordar nociones de SQL y aprender algunas nuevas.

Duración aproximada fue de 140 h

3. Implementación del modelo y validación

Sin duda, esta ha sido la parte más ardua y la que más tiempo a llevado. Es un área con la que no había tenido a penas contacto pero, mediante webs, videotutoriales y la ayuda de un investigador recomendado por la tutora, he ido aprendiendo y se consiguió sacar adelante.

La mayor parte del tiempo invertido ha sido buscando características que añadir al modelo, aprendiendo a manejar los algoritmos y el tiempo de espera para obtener los resultados.

Duración aproximada 300 h

4. Memoria

Por último, la redacción de la memoria.

Duración aproximada 90 h

Costes

Para calcular el coste del trabajo hemos tenido en cuenta por un lado, los costes de *software* y *hardware* y por otro, el coste del personal.

Costes *software* y *hardware*

Los componentes de *software* y *hardware* utilizados han sido los siguientes:

Nombre	Coste
MSI-GL62M-7REX	2050€
Python	0€
Microsoft Office 365	0€
HeidiSQL	0€
XAMPP	0€
Total	2050€

Tabla 2. Costes software y hardware. Elaboración propia

Coste personal

Suponiendo que un ingeniero biomédico cobra 40€ la hora, el coste de la realización del trabajo sería de:

$$\text{Coste personal: } 660 \text{ horas} * 40 \frac{\text{€}}{\text{horas}} = 26.400\text{€}$$

Coste total

Por tanto, el coste total del proyecto sería de:

$$\text{Coste total: } 2050\text{€} + 26400\text{€} = 28450\text{€}$$

Conclusiones

Con este proyecto hemos conseguido obtener un sistema que predice nuevas interacciones de un principio activo, desde un punto de vista del análisis de la semántica de las palabras y haciendo uso del procesamiento natural, una tecnología muy reciente y con mucho potencial presente y futuro.

Se ha planteado desde una perspectiva diferente a los proyectos o investigaciones que se han encontrado, dado que estos se han realizado desde una posición bioquímica.

Entre las dificultades que nos hemos ido encontrando a lo largo del trabajo, ha destacado la poca información relativa a las interacciones positivas o neutras, ya que es normal que lo más interesante sea hallar las negativas para evitar que se produzcan y tenerlas controladas.

Deo mismo modo, las relacionadas con las técnicas de aprendizaje automático, debido a que nunca había hecho uso se ellas y ha supuesto un reto, pero ha sido interesante aprender a usarlas y seguro me ha quedado mucho donde indagar puesto que prácticamente ha sido una toma de contacto.

Implementaciones futuras

La realización del trabajo se ha ajustado a unos tiempos y circunstancias por lo que hay funciones extras que se podrían añadir en un futuro para hacerlo más completo.

Algunas implementaciones para mejorar el sistema que se podrán añadir serían:

- Nuevas interacciones que vayan apareciendo para tener más muestras.
- Poder añadir las características de los principios activos de manera automática.
- Implementar una interfaz gráfica para que sea accesible para todo el mundo.

Bibliografía

1. A. Ordaz, S. Pérez, V. Oliveres (4 de marzo de 2022). *España, cada vez más medicada*. elDiario. https://www.eldiario.es/sociedad/espana-vez-medicada_1_8796419.html
2. A. Ordaz, S. Pérez, V. Oliveres (4 de marzo de 2022). *Consumo medicamentos en España* [Gráfica]. Recuperado de https://www.eldiario.es/sociedad/espana-vez-medicada_1_8796419.html
2. Enciclopedia Médica A.D.A.M. [Internet]. Johns Creek (GA): Ebix, Inc., A.D.A.M.; ©1997-2020. *Tomar múltiples medicamentos de manera segura*; [actualizado 13 ago. 2020; consulta 15 abr. 2022]; [aprox. 4 p.]. Disponible en: <https://medlineplus.gov/spanish/ency/patientinstructions/000883.htm>
3. Uniteco Profesional. (22 de junio de 2021) *Reacciones Adversas a medicamentos: cómo debe actuar un profesional*. <https://www.unitecoprofesional.es/blog/reacciones-adversas-a-medicamentos/>
4. *Reacciones adversas de los medicamentos*. J. Formigos, 33630- Farmacologías y principios de bioquímica. Universidad de Alicante. https://rua.ua.es/dspace/bitstream/10045/20662/1/TEMA_3.pdf
5. Interacción farmacológica. (2021, 3 de noviembre). *Wikipedia, La enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Interacci%C3%B3n_farmacol%C3%B3gica&oldid=139469535.
6. Shalini S. Lynch, PharmD, University of California San Francisco School of Pharmacy (2019). *Interacciones farmacológicas*. Manual MSD versión para profesionales. <https://www.msdmanuals.com/es/professional/farmacolog%C3%ADa-cl%C3%ADnica/factores-que-afectan-la-respuesta-a-los-f%C3%A1rmacos/interacciones-farmacol%C3%B3gicas#:~:text=Una%20interacci%C3%B3n%20f%C3%A1rmaco%20d%C3%A1rmaco%20puede,efectos%20adversos%20o%20fracaso%20terap%C3%A9utico>
7. Meneses, Dr. A. (17 de junio de 2020). *Reacciones provocadas por fármacos*. Salusplay. <https://www.salusplay.com/blog/reacciones-adversas-interacciones-intoxicaciones-farmacos/>
8. Rodríguez Duque, Raisa, Gómez Leyva, Berlis, Rodríguez Moldón, Yarimi, & Díaz Armas, María Teresa. (2019). *Las reacciones adversas como causa de hospitalización*. *Correo Científico Médico*, 23(1), 223-244. Recuperado en 18 de abril de 2022, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1560-43812019000100223
9. *Ley 29/2006, de 26 de julio, de garantías y uso racional de los medicamentos y productos sanitarios*. Boletín Oficial del Estado. 26 de julio de 2006. Consultado el 18 de abril de 2022. <https://www.boe.es/buscar/pdf/2006/BOE-A-2006-13554-consolidado.pdf>
10. Iberdrola. *¿Qué es el procesamiento de lenguaje natural y qué aplicaciones tiene?*. <https://www.iberdrola.com/innovacion/procesamiento-lenguaje-natural-pln>

11. Instituto de ingeniería del conocimiento. *Procesamiento del lenguaje natural*. <https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/>
12. Análisis de sentimiento. Qué es y cómo realizarlo. *QuestionPro*. <https://www.questionpro.com/blog/es/herramienta-de-analisis-de-sentimientos/>
13. Gottlieb, Assaf, Stein, Gideon Y, Oron, Yoram, Ruppín, Eytan, Sharan, Roded, (2012) INDI: a computational framework for inferring drug interactions and their associated recommendations. *Molecular Systems Biology*, 8. 592. doi: accession:10.1038/msb.2012.26 <https://doi.org/10.1038/msb.2012.26>
14. Zhang, P., Wang, F., Hu, J. et al. Label Propagation Prediction of Drug-Drug Interactions Based on Clinical Side Effects. *Sci Rep* 5, 12339 (2015). <https://doi.org/10.1038/srep12339>
15. Foucquier, J., Guedj, M.. Analysis of drug combinations: current methodological landscape, *Pharma Res Per*, 3(3), 2015, e00149, doi: <https://doi.org/10.1002/prp2.149>
16. Huang R-y, Pei L, Liu Q, Chen S, Dou H, Shu G, Yuan Z-x, Lin J, Peng G, Zhang W and Fu H (2019) Isobologram Analysis: A Comprehensive Review of Methodology and Current Research. *Front. Pharmacol.* 10:1222. doi: <https://doi.org/10.3389/fphar.2019.01222>
17. Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, Assempour N, Iynkkaran I, Liu Y, Maciejewski A, Gale N, Wilson A, Chin L, Cummings R, Le D, Pon A, Knox C, Wilson M. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Res.* 2017 Nov 8. <https://go.drugbank.com/> doi: 10.1093/nar/gkx1037.
18. XAMPP. (21 abril de 2022). En *Wikipedia*. <https://es.wikipedia.org/w/index.php?title=XAMPP&oldid=143042775>
19. XAMPP. <https://www.apachefriends.org/es/index.html>
20. HeidiSQL. <https://www.heidisql.com/>
21. Girona Brumós, L. (2020). *Interacciones Farmacológicas*. (Segunda Edición). https://www.sefh.es/bibliotecavirtual/inter20/interacciones_farmacologicas.pdf?ts=20210216191622
22. Agencia Española de Medicamentos y Productos Sanitarios. (s.f). *CIMA REST API v1.19*. Recuperado el 26/04/2022, de https://sede.aemps.gob.es/docs/CIMA-REST-API_1_19.pdf
23. Abellón Ruiz, J. Subiela García, J.A. Díaz Martínez, A.M. (2015). Compatibilidad entre fármacos por vía intravenosa. *Revista Enfermería Docente*. URL: <http://www.sspa.juntadeandalucia.es/servicioandaluzdesalud/huvvsites/default/files/revistas/ED-103-06.pdf>
24. NLTK. (20 febrero de 2022). En *Wikipedia*. <https://es.wikipedia.org/w/index.php?title=NLTK&oldid=141812030>

25. Sckit-learn. (8 noviembre de 2020). En *Wikipedia*. <https://es.wikipedia.org/w/index.php?title=Scikit-learn&oldid=130746122>
26. Pandas. (26 enero de 2022). En *Wikipedia*. [https://es.wikipedia.org/w/index.php?title=Pandas_\(software\)&oldid=141239349](https://es.wikipedia.org/w/index.php?title=Pandas_(software)&oldid=141239349)
27. Funcionamiento TF-IDF.[Imagen]. *El proceso de cálculo de TF-IDF en TfidfVectorizer en sklearn*. Fuente: <https://programmerclick.com/article/70301137743/>
28. Core Radiológico.(2011). Prefijos y sufijos útiles en medicamentos [Fotografía]. URL: <http://www.coreradiologico.com/contenidos/zonas/detalleApartado.aspx?idAPadre=xx0FI7%2FcbVfwa0MvdU%2FWSA%3D%3D&nZ=0IYI6igwXti4EYHtztifxAceLZ7iNI6j%2Fy6jEFWlx%2Fw%3D&idCategoria=khDpqklymopvK4E5qdO8Zg%3D%3D&idApartado=vc9IqYvIaFrl2QWeVnwY%2BA%3D%3D>
29. Andrade, Frank.[Frank Andrade].(2021, 16 de mayo). *Curso completo de machine learning con Scikit-Learn en Python (Clasificación de textos)*. [Video]. YouTube. <https://www.youtube.com/watch?v=V4ab6qsJZMY&list=PLEnAlfUR3oGf4QAJYUmbNxHrmv18HcpNx&index=12>
30. Tutoriales Programación Ya. <https://www.tutorialesprogramacionya.com/>
31. JC-Mouse. (25 abril 2017). Tutorial HeidiSQL: Sesión, Base de datos y tablas. <https://www.jc-mouse.net/base-de-datos/tutorial-heidisql-sesion-base-de-datos-y-tablas>
32. *Python-Requests*. Request. <https://docs.python-requests.org/es/latest/user/quickstart.html#realizar-un-peticion>
33. W3schools.com. 2017. W3Schools Online Web Tutorials. [online] Available at: <https://www.w3schools.com/python/default.asp>
34. Drugs.com. Fuente: <https://www.drugs.com/>
35. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Fuente: <https://scikit-learn.org/stable/modules/classes.html?highlight=svm#module-sklearn.svm>

Anexos

Anexo1

```
import json

import requests

import mysql.connector

'''Funcion mediante la cual añadimos automaticamente a la base de
datos el nombre de los medicamentos obtenidos, los principios
activos y la composicion de los medicamentos'''

def insertarMedicamentos(x,y):

    conexion1=mysql.connector.connect(host="localhost",user="root",pas
swd="",database="medicamentos")#Nos conectamos a nuestra base de
datos

    cursor1=conexion1.cursor()

    params = {"practiv1":x,"pagina":y,"autorizados":"1","comerc":"1"}
    #Parametros que se le agregan a la url con el principio activo y
    solo medicamentos autorizados.

    r=requests.get('https://cima.aemps.es/cima/rest/medicamentos',para
ms) #Obtenemos la página deseada y almacenamos el json en la variable
r.

    r1 = json.loads(r.content) #Leemos el json y lo convertimos en un
valor Python en este caso diccionarios.

    print("Pagina:",r1["pagina"])

    if r1["totalFilas"] != 0:

        print("Total Filas:",r1["totalFilas"])

        for z in r1["resultados"]:

            print("Medicamento:",z["nombre"])

            sql="insert                                into
medicamento(id_medicamento,nombre_medicamento)values(%s,%s)"

            #Sentencia sql para añadir nuevos medicamentos

            filas=("",z["nombre"])

            #Tupla con todos los nombres de los medicamentos

            cursor1.execute(sql,filas)#Añadimos los nuevos
medicamentos a la base de datos
```

```

        conexion1.commit()

        sql2=cursor1.lastrowid

        for x in z["vtm"]["nombre"].capitalize().split(sep='+
'):

            print("Principio activo:",x)

            sql="insert                                ignore                into
principios(id_principio_activo,nombre_principio_activo)values(%s,%
s)"

            filas=("",x)

            cursor1.execute(sql,filas)

            conexion1.commit()

            sql="select  id_principio_activo  from  principios
where nombre_principio_activo = %s"

            cursor1.execute(sql,(x,))

            record = cursor1.fetchone()

            for y in record:

                y

                sql="insert                                ignore                into
composicion(id_medicamento,id_principio_activo)values(%s,%s)"

                filas=(sql2,y)

                cursor1.execute(sql,filas)

                conexion1.commit()

    else:

        print("Ya no hay mas resultados")

```

Anexo 2

```
import mysql.connector
```

```

def insertarInteraccion(x,y,z):

conexion1=mysql.connector.connect(host="localhost",user="root",pas
swd="",database="medicamentos")

    cursor1=conexion1.cursor()

    record = cursor1.fetchone()

    sql1="select  id_principio_activo  from  principios  where
nombre_principio_activo = %s"

    "Seleccionamos el id que corresponde con el primer principio activo
que usaremos"

    cursor1.execute(sql1,(x,))

    record = cursor1.fetchone()

    print(record)

    for o in record:

        o

        sql2="select  id_principio_activo  from  principios  where
nombre_principio_activo = %s"

        "Seleccionamos el id que corresponde con el segundo principio activo
que usaremos"

        cursor1.execute(sql1,(y,))

        record2 = cursor1.fetchone()

        print(record2)

        for h in record2:

            h

            sql="insert                                     into
interaccion(id_principio_activo_1,id_principio_activo_2,tipo_inter
accion)values(%s,%s,%s) "

        "Añadimos los dos id con el tipo de interacción que forman a la
tabla interacción"

        filas=(o,h,z)

        cursor1.execute(sql,filas)

        conexion1.commit()

```

Anexo 3

```
'''Funcion medicante la cual obtenemos una lista de los cambios
introducidos en la base de datos de medicamentos de CIMA'''

def actualizacion(x,y):

    conexion1=mysql.connector.connect(host="localhost",user="root",pas
swd="",database="medicamentos")

    cursor1=conexion1.cursor()

    params = {"fecha":x,"pagina":y}#Parametros que se le añaden a la
url, en este caso Fecha a partir de la cual se desea conocer que
medicamentos se han dado de alta

r=requests.get('https://cima.aemps.es/cima/rest/registroCambios',p
arams)

    r1 = json.loads(r.content)

    print("Pagina:",r1["pagina"])#Mostramos el numero de pagina al
que hemos accedido

    if r1["totalFilas"] != 0: #Indicamos que mientras el numero de
filas sea distinto de zero aún quedan paginas por revisar

        print("Total Filas:",r1["totalFilas"])

        for y in r1["resultados"]:

            params = {"nregistro":y["nregistro"]}

            r
            =
requests.get('https://cima.aemps.es/cima/rest/medicamentos',params
)

            r1 = json.loads(r.content)

            for z in r1["resultados"]:

                if y["tipoCambio"] == 1 and
z["vtm"]["nombre"]!="null": #Si se han añadido nuevos medicamentos
se mostraran por pantalla y serán añadidos a la base de datos

                    print("Medicamento:",z["nombre"])

                    sql="insert ignore into
medicamento(id_medicamento,nombre_medicamento)values(%s,%s)"

                    filas=("",z["nombre"])

                    cursor1.execute(sql,filas)
```

```

conexion1.commit()

sql2=cursor1.lastrowid

for x in
z["vtm"]["nombre"].capitalize().split(sep='+ '): #Si el principio
activo que forma a los nuevos medicamentos no lo tenemos registrado
serán añadidos a la tabla principios

    print("Principio activo:",x)

    sql="insert ignore into
principios(id_principio_activo,nombre_principio_activo)values(%s,%
s)"

    filas=("",x)

    cursor1.execute(sql,filas)

    conexion1.commit()

    sql="select id_principio_activo from principios
where nombre_principio_activo = %s"

    cursor1.execute(sql,(x,))

    record = cursor1.fetchone()

    for y in record:

        y

        sql="insert into
composicion(id_medicamento,id_principio_activo)values(%s,%s)"

        filas=(sql2,y)

        cursor1.execute(sql,filas)

        conexion1.commit()

else:

    print("Ya no hay mas resultados")

```

Anexo 4

```
import nltk
import pandas as pd
import csv

from nltk.stem import SnowballStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from csv import reader
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

stop_word = stopwords.words('spanish')

stemmer = SnowballStemmer("spanish", ignore_stopwords=False)

data_1 = pd.read_csv('principios.csv')

data_1['nombre_principio_activo'] =
data_1['nombre_principio_activo'].str.lower()

data_1['nombre_principio_activo'] =
data_1['nombre_principio_activo'].apply(lambda x : stemmer.stem(x))

vocabulario = data_1['nombre_principio_activo'].unique().tolist()

#Proporcionamos el vocabulario para que el modelo tenga todas las
palabras que se puede encontrar y no las puntue como 0 con lo que
no tendria ninguna utilidad
```

#Si no se le proporciona el vocabulario, se determina un vocabulario a partir de los documentos de entrada.

```
data = pd.read_csv('interaccion_nombre_prueba.csv')

data['nombre_principio_1'] = data['nombre_principio_1'].str.lower()

data['nombre_principio_1'] = data['nombre_principio_1'].apply(lambda x : stemmer.stem(x))

data['nombre_principio_2'] = data['nombre_principio_2'].str.lower()

data['nombre_principio_2'] = data['nombre_principio_2'].apply(lambda x : stemmer.stem(x))

data['tipo_interaccion'] = data['tipo_interaccion'].str.lower()
```

#Usar los dos en el train no se podía, se tenía que entrenar por separado y luego multiplicar

```
train, test = train_test_split(data, test_size=0.20, random_state=0)

train_x_1, train_y = train['nombre_principio_1'], train['tipo_interaccion']

test_x_1, test_y = test['nombre_principio_1'], test['tipo_interaccion']

train_x_2, train_y = train['nombre_principio_2'], train['tipo_interaccion']

test_x_2, test_y = test['nombre_principio_2'], test['tipo_interaccion']
```

```
tfidf = TfidfVectorizer(stop_words=stop_word, vocabulary = vocabulario)
```

```
train_x_3 = tfidf.fit_transform(train_x_1)
```

```
test_x_3 = tfidf.transform(test_x_1)
```

```
train_x_4 = tfidf.transform(train_x_2)
```

```
test_x_4 = tfidf.transform(test_x_2)
```

```
train_x_vector = train_x_3.multiply(train_x_4)
```

```
test_x_vector = test_x_3.multiply(test_x_4)
```

```

"""Suport Vector Machine(SVM)"""

svc = SVC(kernel='linear',class_weight="balanced")

svc.fit(train_x_vector, train_y)


"""Decision Tree"""

dec_tree = DecisionTreeClassifier(class_weight="balanced")

dec_tree.fit(train_x_vector, train_y)


"""Logistic Regression"""

lr = LogisticRegression(class_weight="balanced")

lr.fit(train_x_vector, train_y)


#Distintas métricas para obtener la precisión de los algoritmos

"""Score (Acuraccy)"""

print("Score (Acuraccy)")

print("SVC: ", svc.score(test_x_vector,test_y))

print("DEC: ", dec_tree.score(test_x_vector,test_y))

print("LR: ", lr.score(test_x_vector,test_y))


"""F1 Score"""

"""F1 Score = 2*(Recall*Precision)/(Recall+Precision)"""

print("F1 Score")

print("SVC: ", f1_score(test_y,
svc.predict(test_x_vector),labels=['positiva','negativa','neutro'],
average=None))

print("DEC: ", f1_score(test_y,
dec_tree.predict(test_x_vector),labels=['positiva','negativa','neutro'],
average=None))

print("LR: ", f1_score(test_y,
lr.predict(test_x_vector),labels=['positiva','negativa','neutro'],
average=None))


"""Reporte Clasificacion"""

```



```

print("Reporte Clasificacion")

print("SVC:          ", classification_report(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro']
))

print("DEC:          ", classification_report(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro']))

print("LR:           ", classification_report(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'])
)

"""Confusion Matrix"""

print("Confusion Matrix")

print("SVC:          ", confusion_matrix(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro']
))

print("DEC:          ", confusion_matrix(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro']))

print("LR:           ", confusion_matrix(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'])
)

```

Anexo 5

```

import nltk

import pandas as pd

import csv

from nltk.stem import SnowballStemmer

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from nltk.corpus import stopwords

from csv import reader

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive_bayes import GaussianNB

```

```

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

data_1 = pd.read_csv('principios_terminaciones_1.csv')
vocabulario = data_1['terminaciones'].unique().tolist()

data = pd.read_csv('interaccion_terminaciones.csv', sep=';')
data['nombre_principio_1'] = data['nombre_principio_1'].str.lower()
data['terminacion_1'] = data['terminacion_1'].str.lower()
data['nombre_principio_2'] = data['nombre_principio_2'].str.lower()
data['terminacion_2'] = data['terminacion_2'].str.lower()
data['tipo_interaccion'] = data['tipo_interaccion'].str.lower()

data['Concat_1'] = data['nombre_principio_1'] + " " + data['terminacion_1']
data['Concat_2'] = data['nombre_principio_2'] + " " + data['terminacion_2']

train, test = train_test_split(data, test_size=0.20, random_state=0)

train_x_1, train_y = train['Concat_1'], train['tipo_interaccion']
test_x_1, test_y = test['Concat_1'], test['tipo_interaccion']

train_x_2, train_y = train['Concat_2'], train['tipo_interaccion']
test_x_2, test_y = test['Concat_2'], test['tipo_interaccion']

tfidf = TfidfVectorizer(vocabulary = vocabulario) #Aquí se le proporciona el vocabulario
train_x_3 = tfidf.fit_transform(train_x_1)
test_x_3 = tfidf.transform(test_x_1)
train_x_4 = tfidf.transform(train_x_2)

```

```

test_x_4 = tfidf.transform(test_x_2)

train_x_vector = train_x_3.multiply(train_x_4)
test_x_vector = test_x_3.multiply(test_x_4)

print("He llegado hasta aqui")

#Suport Vector Machine(SVM)
svc = SVC(kernel='linear',class_weight="balanced")
svc.fit(train_x_vector, train_y)

#Decision Tree
dec_tree = DecisionTreeClassifier(class_weight="balanced")
dec_tree.fit(train_x_vector, train_y)

#Logistic Regression
lr = LogisticRegression(class_weight="balanced")
lr.fit(train_x_vector, train_y)

#Distintas métricas para obtener la precisión de los algoritmos

#Score (Acuraccy)
print("Score (Acuraccy)")
print("SVC: ", svc.score(test_x_vector,test_y))
print("DEC: ", dec_tree.score(test_x_vector,test_y))
print("LR: ", lr.score(test_x_vector,test_y))

#F1 Score
#F1 Score = 2*(Recall*Precision)/(Recall+Precision)

print("F1 Score")

```

```

print("SVC:                ", f1_score(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
, average=None))

print("DEC:                ", f1_score(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
, average=None))

print("LR:                 ", f1_score(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
, average=None))

#Reporte Clasificacion

print("Reporte Clasificacion")

print("SVC:                ", classification_report(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

print("DEC:                ", classification_report(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

print("LR:                 ", classification_report(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

#Confusion Matrix

print("Confusion Matrix")

print("SVC:                ", confusion_matrix(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

print("DEC:                ", confusion_matrix(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

print("LR:                 ", confusion_matrix(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
))

```

Anexo 6

```

import nltk

from nltk.corpus import stopwords

```

```

import re
import pandas as pd
from csv import reader
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
import string
import csv
from nltk.stem import SnowballStemmer
from joblib import dump, load
import pickle

data_1 = pd.read_csv('principios.csv')
vocabulario = data_1['nombre_principio_activo'].unique().tolist()

stop_word = stopwords.words('english')

data = pd.read_csv('principios_caracteristicas_1.csv', sep=';')

data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()
data['Summary'] = data['Summary'].str.lower()
data['Summary'] = data['Summary'].str.replace('[^\w\s]','', regex=True)

```

```

data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word for
word in x.split() if word not in (stop_word)]))

data['Background'] = data['Background'].str.lower()

data['Background'] = data['Background'].str.replace('[^\w\s]',' ',
regex=True)

data['Background'] = data['Background'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Indication'] = data['Indication'].str.lower()

data['Indication'] = data['Indication'].str.replace('[^\w\s]',' ',
regex=True)

data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()

data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]',' ',
regex=True)

data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()

data['Mechanism_of_action'] = data['Mechanism_of_action'].str.replace('[^\w\s]',' ',
regex=True)

data['Mechanism_of_action'] = data['Mechanism_of_action'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['terminacion'] = data['terminacion'].str.lower()


data['Concat'] = data['nombre_principio_activo'] + " " +
data['Summary'] + " " + data['Background'] + " " + data['Indication']
+ " " + data['Pharmacodynamics'] + " " + data['Mechanism_of_action']
+ " " + data['terminacion']


comdic = {}

vocabulario = set()


for row in data.loc[:, ['nombre_principio_activo',
'Concat']].itertuples():

```

```

comdic[row[1]] = row[2]

for w in row[2].split(' '):
    vocabulario.add(w)

data = pd.read_csv('interaccion_terminaciones.csv', sep=';')
data['nombre_principio_1'] = data['nombre_principio_1'].str.lower()
data['terminacion_1'] = data['terminacion_1'].str.lower()
data['nombre_principio_2'] = data['nombre_principio_2'].str.lower()
data['terminacion_2'] = data['terminacion_2'].str.lower()
data['tipo_interaccion'] = data['tipo_interaccion'].str.lower()

data['Concat_1'] = data['nombre_principio_1'].apply(lambda x:
comdic[x])
data['Concat_2'] = data['nombre_principio_2'].apply(lambda x:
comdic[x])

train, test = train_test_split(data, test_size=0.20, random_state=0)

train_x_1, train_y = train['Concat_1'], train['tipo_interaccion']
test_x_1, test_y = test['Concat_1'], test['tipo_interaccion']

train_x_2, train_y = train['Concat_2'], train['tipo_interaccion']
test_x_2, test_y = test['Concat_2'], test['tipo_interaccion']

print("Calculando TF-IDF...")

tfidf = TfidfVectorizer(vocabulary = vocabulario) #Aquí se le
proporciona el vocabulario

train_x_3 = tfidf.fit_transform(train_x_1)
test_x_3 = tfidf.transform(test_x_1)

```

```

train_x_4 = tfidf.transform(train_x_2)
test_x_4 = tfidf.transform(test_x_2)
print("Calculando TF-IDF. Hecho!")

train_x_vector = train_x_3.multiply(train_x_4)
test_x_vector = test_x_3.multiply(test_x_4)

print("He llegado hasta aqui")

filename = 'tfidf_todo.pkl'
pickle.dump(tfidf,open(filename,'wb'))

dump(tfidf, 'tfidf.joblib')

#Suport Vector Machine(SVM)
svc = SVC(kernel='linear',class_weight="balanced")
svc.fit(train_x_vector, train_y)

filename = 'svc_todo.pkl'
pickle.dump(svc,open(filename,'wb'))

dump(svc, 'svc_todo.joblib')

print("He llegado hasta aqui")

#Decision Tree
dec_tree = DecisionTreeClassifier(class_weight="balanced")
dec_tree.fit(train_x_vector, train_y)

filename = 'dec_tree_todo.pkl'
pickle.dump(dec_tree,open(filename,'wb'))

```



```

dump(dec_tree, 'dec_tree_todo.joblib')

print("He llegado hasta aqui")

#Logistic Regression
lr = LogisticRegression(class_weight="balanced")
lr.fit(train_x_vector, train_y)

filename = 'lr_todo.pkl'
pickle.dump(lr, open(filename, 'wb'))

dump(lr, 'lr_todo.joblib')

print("He llegado hasta aqui")

#Distintas métricas para obtener la precisión de los algoritmos

#Score (Acuraccy)
print("Score (Acuraccy)")
print("SVC: ", svc.score(test_x_vector, test_y))
print("DEC: ", dec_tree.score(test_x_vector, test_y))
print("LR: ", lr.score(test_x_vector, test_y))

#F1 Score
#F1 Score = 2*(Recall*Precision)/(Recall+Precision)

print("F1 Score")

print("SVC: ", f1_score(test_y,
svc.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
average=None))

print("DEC: ", f1_score(test_y,
dec_tree.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
average=None))

print("LR: ", f1_score(test_y,
lr.predict(test_x_vector), labels=['positiva', 'negativa', 'neutro'],
average=None))

```

```

#Reporte Clasificacion

print("Reporte Clasificacion")

print("SVC:          ",          classification_report(test_y,
svc.predict(test_x_vector),labels=['positiva','negativa','neutro']
))

print("DEC:          ",          classification_report(test_y,
dec_tree.predict(test_x_vector),labels=['positiva','negativa','neu
tro']))

print("LR:           ",          classification_report(test_y,
lr.predict(test_x_vector),labels=['positiva','negativa','neutro'])
)


#Confusion Matrix

print("Confusion Matrix")

print("SVC:          ",          confusion_matrix(test_y,
svc.predict(test_x_vector),labels=['positiva','negativa','neutro']
))

print("DEC:          ",          confusion_matrix(test_y,
dec_tree.predict(test_x_vector),labels=['positiva','negativa','neu
tro']))

print("LR:           ",          confusion_matrix(test_y,
lr.predict(test_x_vector),labels=['positiva','negativa','neutro'])
)

```

Anexo 7

```

import pickle

import pandas as pd

from nltk.corpus import stopwords

def load_model(rutaModelo = "pickle_model.pkl"):

```

```

"""
Lee modelos serializados
"""

with open(rutaModelo, 'rb') as file:
    pickle_model = pickle.load(file)
    return pickle_model

def get_drugname():
    """
    Consultar BD, API, etc.
    """
    return "Loperamida"

def get_description(drugname):
    """
    Devuelve características (de momento es una maqueta)
    """

    stop_word = stopwords.words('english')

    data = pd.read_csv('principios_caracteristicas_1.csv', sep=';', encoding='cp1252')

    data = data[data['nombre_principio_activo']==drugname] # debería ser una sola fila

    data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()

    data['Summary'] = data['Summary'].str.lower()

    data['Summary'] = data['Summary'].str.replace('[^\w\s]', '', regex=True)

```

```

    data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word
for word in x.split() if word not in (stop_word)]))

    data['Background'] = data['Background'].str.lower()

    data['Background'] = data['Background'].str.replace('[^\w\s]',' ',
regex=True)

    data['Background'] = data['Background'].apply(lambda x: ' '.join([word
for word in x.split() if word not in (stop_word)]))

    data['Indication'] = data['Indication'].str.lower()

    data['Indication'] = data['Indication'].str.replace('[^\w\s]',' ',
regex=True)

    data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word
for word in x.split() if word not in (stop_word)]))

    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()

    data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]',' ',
regex=True)

    data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda
x: ' '.join([word for word in x.split() if word not in (stop_word)]))

    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()

    data['Mechanism_of_action'] = data['Mechanism_of_action'].str.replace('[^\w\s]',' ',
regex=True)

    data['Mechanism_of_action'] = data['Mechanism_of_action'].apply(lambda x: ' '.join([word
for word in x.split() if word not in (stop_word)]))

    data['terminacion'] = data['terminacion'].str.lower()


    data['Concat'] = data['nombre_principio_activo'] + " " +
data['Summary'] + " " + data['Background'] + " " + data['Indication']
+ " " + data['Pharmacodynamics'] + " " + data['Mechanism_of_action']
+ " " + data['terminacion']


    return data['Concat'].values[0]


def get_clfdata(drugfeatures, drugname):
    """

```

```

Obtiene la tabla para clasificar

"""

stop_word = stopwords.words('english')

data = pd.read_csv('principios_caracteristicas_1.csv', sep=';', encoding='cp1252')

data['nombre_principio_activo'] = data['nombre_principio_activo'].str.lower()

data['Summary'] = data['Summary'].str.lower()

data['Summary'] = data['Summary'].str.replace('[^\w\s]','', regex=True)

data['Summary'] = data['Summary'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Background'] = data['Background'].str.lower()

data['Background'] = data['Background'].str.replace('[^\w\s]','', regex=True)

data['Background'] = data['Background'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Indication'] = data['Indication'].str.lower()

data['Indication'] = data['Indication'].str.replace('[^\w\s]','', regex=True)

data['Indication'] = data['Indication'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Pharmacodynamics'] = data['Pharmacodynamics'].str.lower()

data['Pharmacodynamics'] = data['Pharmacodynamics'].str.replace('[^\w\s]','', regex=True)

data['Pharmacodynamics'] = data['Pharmacodynamics'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_word)]))

data['Mechanism_of_action'] = data['Mechanism_of_action'].str.lower()

```

```

data['Mechanism_of_action'] =
data['Mechanism_of_action'].str.replace('[^\w\s]',' ',regex=True)

data['Mechanism_of_action'] =
data['Mechanism_of_action'].apply(lambda x: ' '.join([word for word
in x.split() if word not in (stop_word)]))

data['Comp2'] = data['nombre_principio_activo'] + " " +
data['Summary'] + " " + data['Background'] + " " + data['Indication']
+ " " + data['Pharmacodynamics'] + " " + data['Mechanism_of_action']

data['Comp1'] = drugfeatures

data['Comp1Name'] = drugname

data['Comp2Name'] = data['nombre_principio_activo']

return data.loc[:, ['Comp1Name', 'Comp2Name', 'Comp1', 'Comp2']]

```

```

def update_data(drugname):

```

```

    """

```

```

    Obtiene la información relevante (Summary, Background, etc.)

```

```

    """

```

```

    path = 'principios_caracteristicas_1.csv'

```

```

    data = pd.read_csv(path, sep=';')

```

```

    drugdata = { 'id_principio_activo': 15001,

```

```

                'nombre_principio_activo': drugname,

```

```

                'Summary': 'Loperamide is a long acting antidiarrheal
used to control nonspecific diarrhea and chronic diarrhea caused by
inflammatory bowel disease, or gastroenteritis.',

```

```

                'Background': 'One of the long-acting synthetic
antidiarrheals; it is not significantly absorbed from the gut, and
has no effect on the adrenergic system or central nervous system,
but may antagonize histamine and interfere with acetylcholine
release locally.',

```

```

                'Indication': 'For the control and symptomatic relief
of acute nonspecific diarrhea and of chronic diarrhea associated

```

with inflammatory bowel disease or gastroenteritis. Also used for reducing the volume of discharge from ileostomies.',

'Pharmacodynamics': 'Loperamide is a synthetic anti-diarrheal indicated for the control and symptomatic relief of acute nonspecific diarrhea and of chronic diarrhea associated with inflammatory bowel disease. Loperamide is also indicated for reducing the volume of discharge from ileostomies. In man, Loperamide prolongs the transit time of the intestinal contents. It reduces the daily fecal volume, increases the viscosity and bulk density, and diminishes the loss of fluid and electrolytes. Tolerance to the antidiarrheal effect has not been observed. Loperamide is an opioid receptor agonist and acts on the mu opioid receptors in the myenteric plexus large intestines; it does not affect the central nervous system like other opioids. It works specifically by decreasing the activity of the myenteric plexus which decreases the motility of the circular and longitudinal smooth muscles of the intestinal wall. This increases the amount of time substances stay in the intestine, allowing for more water to be absorbed out of the fecal matter. Loperamide also decreases colonic mass movements and suppresses the gastrocolic reflex.',

'Mechanism_of_action': "In vitro and animal studies show that Loperamide acts by slowing intestinal motility and by affecting water and electrolyte movement through the bowel. Loperamide inhibits peristaltic activity by a direct effect on the circular and longitudinal muscles of the intestinal wall. It is a non-selective calcium channel blocker and binds to opioid mu-receptors. Evidence also suggests that at higher concentrations it binds to calmodulin.",

'terminacion': 'amida'

}

data = data.append(drugdata, ignore_index=True)

data.to_csv(path, sep=';')

Anexo 8

```
from production_utils_1 import load_model, get_drugname,
get_description, get_clfdata, update_data
```

```
## preparar datos
```

```
# obtener datos a clasificar
```

```

newdrugname = get_drugname()

# actualizar csv (porque se lee desde ahí ahora...pero puede
cambiarse)
update_data(newdrugname)

# preparar datos (extraer características, etc.)
newdrugdesc= get_description(newdrugname)

# preparar datos para clasificación
clfdata = get_clfdata(newdrugdesc, newdrugname)

## preparar clasificación

# leer modelos
tfidfpath = 'tfidf_todo.pkl'
tfidf = load_model(tfidfpath)

clfpath = 'dec_tree_todo.pkl'
clf = load_model(clfpath)
# aplicar funciones de extracción de características

tfcomp1 = tfidf.transform(clfdata['Comp1'])
tfcomp2 = tfidf.transform(clfdata['Comp2'])
comp1Name = clfdata['Comp1Name']
comp2Name = clfdata['Comp2Name']

features = tfcomp1.multiply(tfcomp2)

# clasificar
pred = clf.predict(features)

```



```
for c1n, c2n, p in zip(comp1Name, comp2Name, pred):  
    print(f'{c1n} - {c2n} -> {p}')
```