Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de un clasificador adaptativo de emergencias en medicina espacial

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Ignacio Garrido Sevilla

Tutor: Juan Miguel García Gómez

Director experimental: Pablo Ferri Borreda

Curso 2020-2021

Resumen

A principios de 2021 tres naves espaciales, el Hope Orbiter, el Tianwen-1 y el Perseverance, llegaron a Marte con la esperanza de llevar a cabo misiones específicas para estudiar su atmósfera y su superficie, con el objetivo concreto de detectar signos claros de vida. Los tres vuelos espaciales son misiones de exploración basadas en la tecnología de los instrumentos y la robótica, pero aún no ha sido la ventana orbital para la exploración humana de Marte.

La adaptación de los sistemas de soporte vital y de salud a las condiciones espaciales de largas misiones de exploración, suponen un reto para las tecnologías aún no resuelto debido a la gran distancia para volver a la Tierra y la latencia de las comunicaciones con control de misión. Específicamente, todavía no se dispone de sistemas de ayuda a la decisión médica que permitan saber si una emergencia sanitaria supone un riesgo vital para la tripulación. En este proyecto hemos desarrollado sistemas de ayuda a la decisión médica que permitan predecir nuevos casos de condiciones médicas comunes en el espacio que, según la NASA, suponen riesgo vital para los tripulantes involucrados en una emergencia espacial.

Para ello, partimos desde el conjunto de las cien condiciones médicas identificadas como de mayor probabilidad de cara a los viajes espaciales identificadas por los investigadores Elkin Romero y David Francisco. A su vez, contamos con una base de datos de más de 130.000 casos clínicos facilitada por el servicio de emergencias del 112 de la Comunidad Valenciana. Sin embargo, veinte de las citadas condiciones son realmente exiguas dentro del repositorio, por lo que hemos llevado a cabo un generador de pacientes virtuales que permita solventar dicho problema. Para lograrlo, ha sido necesario utilizar literatura médica de todas y cada una de las patologías identificadas.

Asimismo, hemos evaluado la bondad de los modelos obtenidos para ambos conjuntos de pacientes de forma desagregada y conjunta, de forma que pudiéramos estudiar tanto el poder clasificador de nuestro sistema como la utilidad real de dichos pacientes virtuales y de los procedimientos empleados. Tras ello, hemos podido ver que la clasificación obtenida es realmente buena, siendo el valor F1 superior a 0,82 en los cinco modelos utilizados. No obstante, también hemos comprobado que algunas de las técnicas del procesamiento del lenguaje, como la «*lemmatization*» o la selección de términos en la matriz TF-IDF no suponen una mejora sustancial.

Además, para facilitar la clasificación de los casos clínicos y la generación de pacientes virtuales hemos elaborado una página web que permita realizar ambas acciones con un simple «click», de forma que mediante cualquier navegador se puedan realizar pruebas sin la necesidad de acceder al código fuente original ni tener que realizar instalación alguna.

Por último, el desarrollo del presente trabajo se realiza con la perspectiva de construir una primera aproximación a un sistema mucho más complejo y robusto que denominaremos MEDEA y que supondría un salto cualitativo en la seguridad de los astronautas de cara la venidera exploración espacial.

Palabras clave: *Ayuda a la decisión médica, medicina espacial, emergencia médica, aprendizaje automático*

Resum

A principis de 2021 tres naus espacials, l'Hope Orbiter, el Tianwen-1 i el Perseverance, van arribar a Mart amb l'esperança de dur a terme missions específiques per a estudiar la seua atmosfera i la seua superfície, amb l'objectiu concret de detectar signes clars de vida. Els tres vols espacials són missions d'exploració basades en la tecnologia dels instruments i la robòtica, però encara no ha sigut la finestra orbital per a l'exploració humana de Mart.

L'adaptació dels sistemes de suport vital i de salut a les condicions espacials de llargues missions d'exploració, suposen un repte per a les tecnologies encara no resolt a causa de la gran distància per a tornar a la Terra i la latència de les comunicacions amb control de missió. Específicament, encara no es disposa de sistemes d'ajuda a la decisió mèdica que permeten saber si una emergència sanitària suposa un risc vital per a la tripulació. En este projecte hem desenrotllat sistemes d'ajuda a la decisió mèdica que permeten predir nous casos de condicions mèdiques comunes en l'espai que, segons la NASA, suposen risc vital per als tripulants involucrats en una emergència espacial.

Per a això, partim des del conjunt de les cent condicions mèdiques identificades com de major probabilitat de cara als viatges espacials identificades pels investigadors Elkin Romero i David Francisco. Al seu torn, comptem amb una base de dades de més de 130.000 casos clínics facilitada pel servici d'emergències del 112 de la Comunitat Valenciana. No obstant això, vint de les esmentades condicions són realment exigües dins del reposador, per la qual cosa hem dut a terme un generador de pacients virtuals que permeta resoldre el dit problema. Per a aconseguir-ho, ha sigut necessari utilitzar literatura mèdica de totes i cada una de les patologies identificades.

Així mateix, hem avaluat la bondat dels models obtinguts per a ambdós conjunts de pacients de forma desagregada i conjunta, de manera que poguérem estudiar tant el poder classificador del nostre sistema com la utilitat real de dites pacients virtuals i dels procediments empleats. Després d'això, hem pogut veure que la classificació obtinguda és realment bona, sent el valor F1 superior a 0,82 en els cinc models utilitzats. No obstant això, també hem comprovat que algunes de les tècniques del processament del llenguatge, com la «lemmatization» o la selecció de termes en la matriu TF-IDF no suposen una millora substancial.

A més, per a facilitar la classificació dels casos clínics i la generació de pacients virtuals hem elaborat una pàgina web que permeta realitzar ambdós accions amb un simple «click», de manera que per mitjà de qualsevol navegador es puguen realitzar proves sense la necessitat d'accedir al codi font original ni haver de realitzar cap instal·lació.

Finalment, el desenrotllament del present treball es realitza amb la perspectiva de construir una primera aproximació a un sistema molt més complex i robust que denominarem MEDEA i que suposaria un salt qualitatiu en la seguretat dels astronautes de cara la venidora exploració espacial.

Palabras clave: *Ajuda a la decisió mèdica, medicina espacial, emergència mèdica, aprenentatge automàtic*

Abstract

In early 2021 three spacecraft, the Hope Orbiter, Tianwen-1 and Perseverance, arrived at Mars in the hope of carrying out specific missions to study its atmosphere and surface, with the specific goal of detecting clear signs of life. All three spaceflights are exploration missions based on instrument technology and robotics, but it has not yet been the orbital window for human exploration of Mars.

Adapting life support and health support systems to the space conditions of long exploration missions suppose a challenge for technologies not yet solved due to the long distance from Earth and the latency of communications with mission control. Specifically, medical decision support systems that would let us know if a health emergency supposes a life-threatening risk to the crew are not yet available. In this project, we have developed medical decision support systems that can predict new cases of common medical conditions in space that, according to NASA, are life-threatening to crewmembers involved in a space emergency.

To do so, we started from the set of the one hundred medical conditions identified by researchers Elkin Romero and David Francisco as most likely to occur during space travel. In turn, we rely on a database of more than 130,000 clinical cases provided by the emergency service of 112 of the Valencian Community. However, twenty of the mentioned conditions are meager within the repository, so we have carried out a virtual patient generator to solve this problem. To achieve this, it has been necessary to use medical literature for each and every one of the pathologies identified.

We have also evaluated the quality of the models obtained for both sets of patients in a disaggregated and joint way, so that we could study both the classifying power of our system and the real utility of these virtual patients and the procedures used.

After this, we have been able to see that the classification obtained is really good, with the F1 value being higher than 0.82 in the five models used. However, we have also found that some of the language processing techniques, such as lemmatization or the selection of terms in the TF-IDF matrix, do not represent a substantial improvement.

In addition, to facilitate the classification of clinical cases and the generation of virtual patients, we have developed a web page that allows both actions to be performed with a simple "click", so that tests can be performed using any browser without the need to access the original source code or to perform any installation.

Finally, the development of this work is carried out with the perspective of building a first approximation to a much more complex and robust system that we will call MEDEA and that would represent a qualitative leap in the safety of astronauts for the coming space exploration.

Key words: *Medical decision support, space medicine, deep learning, adaptive machine learning*

Índice general

Índice de figuras

Índice de tablas

1 Introducción

1.1 Motivación

La exploración espacial por parte de personas presenta dos problemas importantes:

- Las distancias con respecto a la Tierra serán realmente notables conforme se desarrolle la actividad espacial. Ya no se habla sólo del tiempo que se requerirá en realizar el viaje, sino que las comunicaciones en tiempo real serán del todo imposibles. Por ejemplo, debido a la distancia entre el planeta Tierra y Marte y considerando la velocidad de la luz, las comunicaciones entre ambos puntos se pueden demorar desde tres minutos hasta veintidós.

- El envío tanto de personas como de recursos es limitado. Las primeras colonias humanas fuera de la Tierra contarán con pocas personas y será necesario un conjunto de perfiles muy heterogéneos: desde geólogos hasta ingenieros, pasando por botánicos, psicólogos y médicos. Es decir, el número de personas que exista en cada una de las misiones con conocimientos de cada una de las disciplinas que se consideran necesarias en este tipo de misiones será realmente bajo.

Ambas dificultades supondrán un enorme riesgo para la tripulación en situaciones de emergencia, pero, en especial, las emergencias sanitarias supondrán un gran peligro debido a que la cantidad de médicos o especialistas de la salud será limitada y el acceso a los conocimientos de alguien que se encuentre en la Tierra podrían llegar demasiado tarde. La NASA es perfectamente consciente de esta problemática y ya se plantea formas de mitigar sus consecuencias [1].

Como se acaba de mencionar, la salud de los astronautas y la gestión de emergencias sanitarias supone un gran peligro para operaciones fuera de la Tierra, como las futuras misiones tripuladas de SpaceX planeadas para 2026 o las misiones americanas y chinas para la década de 2030, y podría añadir dificultad a la ya de por sí compleja exploración espacial. Por este motivo, se debe diseñar algún mecanismo que facilite o reduzca los riesgos de esta actividad.

En este sentido, la propuesta que se persigue con este proyecto consiste en crear un **sistema de ayuda a la decisión médica para misiones de exploración espacial**. Este software médico sería el encargado de ofrecer asistencia a la tripulación para la toma de decisiones en caso de emergencias gracias al uso de Inteligencia Artificial. De esta forma, la aplicación pondría el foco no en el diagnóstico, sino en la **urgencia de la atención al paciente** y el **riesgo vital que su**

situación conlleva. Para ello, el sistema deberá trabajar bajo una serie de condiciones que le permitan alcanzar su objetivo:

- **Tiempo real**: la información que proporcione dicho sistema debe ofrecerse en tiempo real, pues su objetivo es ejercer como apoyo a personas en situaciones de riesgo donde la capacidad de demora suele ser baja o inexistente.

- **Autonomía**: el sistema debe ser totalmente autónomo, aunque respete la decisión humana, capaz de ofrecer una respuesta sin necesitar la intermediación de un ser humano para generarla ni la comunicación con la Tierra u otros sistemas remotos inalcanzables en ese momento por el retardo entre las comunicaciones.

- **Adaptabilidad**: las decisiones no serán de carácter estático, sino que dependerán del contexto. Por ello, el sistema creado debe ser capaz de adaptar sus predicciones a las nuevas condiciones dadas de forma constante.

- **Variables de entrada**: este software no deberá basar sus predicciones exclusivamente en reglas o datos históricos, sino que deberá estar alimentado con la información de soporte vital de los miembros de la tripulación y utilizar esta monitorización para sus recomendaciones.

- **Consciencia del entorno**: las recomendaciones facilitadas deberán tener en cuenta el equipamiento y los suministros en cada momento. No tendría sentido tomar decisiones médicas bajo un supuesto totalmente falso como el asumir la disponibilidad de todos los recursos en cantidades infinitas. La escasez de material será un hecho y, por tanto, se debe tomar en consideración.

- **Optimización**: en base a todo lo anterior, se observa que el sistema planteado debe atender a una serie de precondiciones según las circunstancias. Es decir, debe ofrecer asistencia en la toma de decisiones por parte de personas teniendo una serie de prioridades que deberá optimizar (como la minimización de riesgos o de uso de recursos) para alcanzar la solución más satisfactoria.

- **Estándares**: el sistema desarrollado deberá cumplir con estándares médicos de cuidado nivel V[1] y éticos para vuelos espaciales e Inteligencia Artificial.

- **Aplicado**: debemos tener en cuenta que tanto las patologías como su frecuencia de aparición y los síntomas asociados serán distintos en el espacio exterior. Por ello es necesario diseñar un sistema aplicado a estas circunstancias que, en el futuro, se nutra de los casos y condiciones propios a dicho entorno hostil conforme se incrementen los viajes de exploración espacial.

De esta forma, un software que cumpliera con todas las características que han sido mencionadas, podría llegar a ser un elemento realmente transformador.

[1] Entendido como el conjunto de actividades destinadas promoción de la salud, la prevención de enfermedades, la curación, la rehabilitación de los pacientes y los cuidados paliativos.

1.2 Objetivos

El objetivo principal del presente Trabajo Fin de Grado es **definir, diseñar y evaluar** una posible primera implementación de un **sistema de clasificación de riesgo vital de emergencias sanitarias**, especializado para los **viajes de exploración espacial**. De forma más concreta, deberá ser capaz de establecer, mediante el análisis de **texto libre**, la **existencia o no de riesgo** vital en un caso clínico. En línea con la definición SMART de objetivos, podemos concretar cinco más específicos:

- Diseñar un sistema capaz de obtener, de forma automática, información suficiente para cada una de las condiciones médicas a estudiar, siendo ésta suficiente para crear casos clínicos virtuales a partir de la misma.
- Definir un algoritmo capaz de crear pacientes virtuales clasificados por CIE (patología médica) y por existencia de riesgo o no de forma automática y configurable.
- Entrenar distintos modelos utilizando casos reales y los pacientes virtuales desarrollados previamente de forma conjunta y separada, analizando las diferencias obtenidas en los resultados.
- Lograr un sistema clasificador capaz de determinar la existencia de riesgo o no con una precisión superior al 80% utilizando un conjunto de pacientes diseñados, de forma específica, parar simular casos en el espacio.

1.3 Guía resumen de la memoria

- **Capítulo 2**: Aproximación teórica a los contenidos relacionados con el presente proyecto. Además, se desarrollará el estado del arte de las tecnologías afines.
- **Capítulo 3**: Descripción de los elementos utilizados para el trabajo, desde el equipamiento hardware y software hasta los datos y sus fuentes.
- **Capítulo 4**: Se esgrimirán los procedimientos empleados a lo largo del trabajo, tanto para la generación de los pacientes virtuales como de los modelos clasificadores.
- **Capítulo 5**: Tras realizar los experimentos, se expondrán los resultados obtenidos en cada uno de ellos y se realizará un breve análisis, llevando a cabo comparaciones entre modelos y experimentos distintos.
- **Capítulo 6**: Conclusiones extraídas en base a los resultados previos y al análisis realizado en el Capítulo anterior, así como a la investigación efectuada durante el trabajo.
- **Capítulo 7**: Se expondrá una visión de las líneas futuras de este proyecto enmarcadas dentro del desarrollo de un sistema complejo denominado MEDEA.

Por último, cabe destacar que los ficheros y bases de datos de los que se hace mención a lo largo del trabajo se pueden encontrar en el repositorio creado para tal efecto, accesible desde el

siguiente enlace: [https://github.com/igarridosevilla/tfg-garrido-sevilla]. No obstante, todos aquellos archivos que contienen algún caso clínico extraído de la base de datos de las llamadas al 112 de la Comunidad Valenciana han sido omitidos debido al contrato de confidencialidad firmado con la Generalitat Valenciana. Asimismo, los modelos generados tampoco están incluidos por su excesivo peso (superior a los cien megabytes en algún caso).

2 Estado del arte

En el segundo capítulo del presente documento se realizará una revisión de la situación actual de las tecnologías empleadas y de los conceptos relacionados con el proyecto, además de definir los conceptos necesarios para el desarrollo del mismo.

2.1 Inteligencia artificial y Machine Learning

Es notable la creciente popularidad que están adquiriendo términos como Inteligencia Artificial y Machine Learning. En muchas ocasiones, esto términos se usan de forma indistinta y, a pesar de que parezca que se refieren a lo mismo, es importante matizar las diferencias.

Según la consultora *Mckinsey*, la **Inteligencia Artificial** (**AI** por sus siglas en inglés) se define como «*the ability of a machine to perform cognitive functions we associate with human minds, such as perceiving, reasoning, learning, interacting with the environment, problem solving, and even exercising creativity*» [2]. Es decir, cuando se habla de la AI, se engloba el conjunto de metodologías que tratan, desde una perspectiva teórica y práctica, el estudio, el desarrollo y la aplicación de técnicas informáticas que permitan a las máquinas imitar algunos aspectos de la inteligencia humana.

Así pues, cuando se habla del **Machine Learning** (**ML**) se está haciendo referencia al conjunto de algoritmos que tratan de analizar los datos, aprender de ellos, encontrar patrones y tomar decisiones informadas basándose en los resultados obtenidos en lugar de basarse en una programación previa explícita [2]. Por tanto, tal y como se puede observar en la Figura 1, el ML es una subcategoría que se encuentra englobada bajo el paraguas de la inteligencia artificial.

A su vez, dentro del ML se puede desagregar otra categoría más, la de **Deep Learning** (**DL**). Este conjunto relativamente nuevo de categorías dentro del ML hace referencia a modelos de nueva generación, como Redes Neuronales Convolucionales (CNN) o las Redes Generativas Antagónicas (GAN), que permiten explotar relaciones intrínsecas entre cantidades masivas de datos, con menos interacción humana en su preprocesado y que están obteniendo resultados excelentes en tareas hasta ahora no resultas, como la detección de objetos parciales en imágenes. Así pues, estas nuevas técnicas lideran actualmente las tareas de clasificación de imágenes y reconocimiento facial, entre otros.

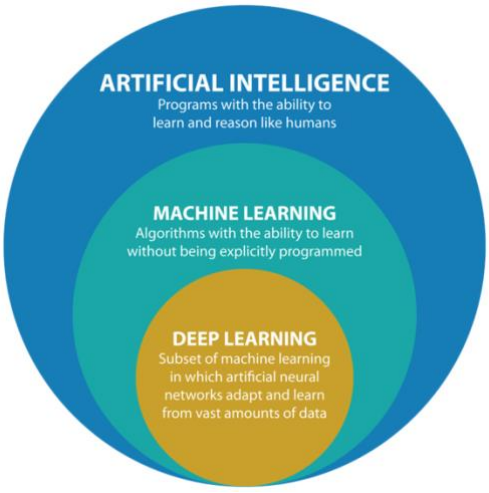

Figura 1 *Jerarquía dentro de la AI*
Fuente: [3]

Una vez realizada esta matización es conveniente recalcar también que el ML no es una disciplina única y que se compone de distintos tipos de aprendizaje y diferentes algoritmos. Es decir, aunque todas las técnicas englobadas en este concepto basan su funcionamiento en el uso de datos como entrada para generar patrones y conocimiento que permitan conocer o predecir aspectos de la materia estudiada, en función de la estructura de estos datos de entrada podemos diferenciar, principalmente, dos tipos de aprendizaje:

- **Aprendizaje supervisado:** Este primer tipo de aprendizaje es el utilizado cuando los datos empleados se componen de los datos como tal y de una variable de control o *feedback*, generalmente llamada *etiqueta*. Es decir, los elementos a analizar se encuentran etiquetados, por lo que se utilizan para aprender la forma de predecir o clasificar nuevos datos. Es este el motivo por el que adquiere la condición de supervisado, puesto que demanda de la supervisión del ser humano para que, previamente, sean definidas las etiquetas a utilizar.

- **Aprendizaje no supervisado:** Este segundo paradigma hace referencia a aquellos datos que no se encuentran etiquetados de ninguna forma. Por tanto, en estos casos se busca que las técnicas de Machine Learning busquen e identifiquen patrones de forma autónoma. En este caso es evidente que la intervención humana es totalmente innecesaria, puesto que no se requiere que los datos se encuentren etiquetados y es la propia máquina quien se encarga de descubrir los patrones por sí misma.

En el caso de este proyecto, en el que el objetivo final es obtener un sistema capaz de realizar una clasificación binaria de riesgo vital en emergencias médicas (SÍ o NO) y los datos a emplear ya se encuentran etiquetados, se trabajará de forma exclusiva con el aprendizaje supervisado. Por tanto, de todos los diferentes modelos que existen dentro de este tipo de aprendizaje con la capacidad de predecir o clasificar datos, a continuación, se destacarán aquellos que permiten alcanzar la segunda meta.

2.1.1 Naive Bayes

Este algoritmo de aprendizaje automático es un clasificador probabilístico que se basa, fundamentalmente, en el **Teorema de Bayes**. Este principio, el cual podemos ver bajo este párrafo, expresa la probabilidad condicionada de un suceso A dado otro suceso B en función de sus probabilidades a priori p(A) y p(B) y la función de probabilidad condicionada de B dado A.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

A su vez el concepto de *naive* o inocente se debe a que asume que todas las variables utilizadas son independientes entre sí, es decir, que la presencia de una característica concreta en un conjunto de datos no está relacionada con la presencia de cualquier otra distinta.

Aunque esta precondición es difícil de cumplir en la realidad, hace posible forjar modelos de una forma rápida y sencilla. Su uso destaca en los problemas de clasificación binarios y multiclase. Por todo ello, aunque no se asemeja al funcionamiento de los datos en el mundo real, sigue siendo un modelo a tener en cuenta por su simplicidad.

2.1.2 Random Forest

Otro algoritmo ampliamente conocido y utilizado dentro del Machine Learning son los **árboles de decisión**. Como vemos en la Figura 2, este modelo se basa en la generación de nodos de separación de casos a partir de distintas condiciones para, finalmente, ofrecer un output en función del nodo-hoja al que se llegue.

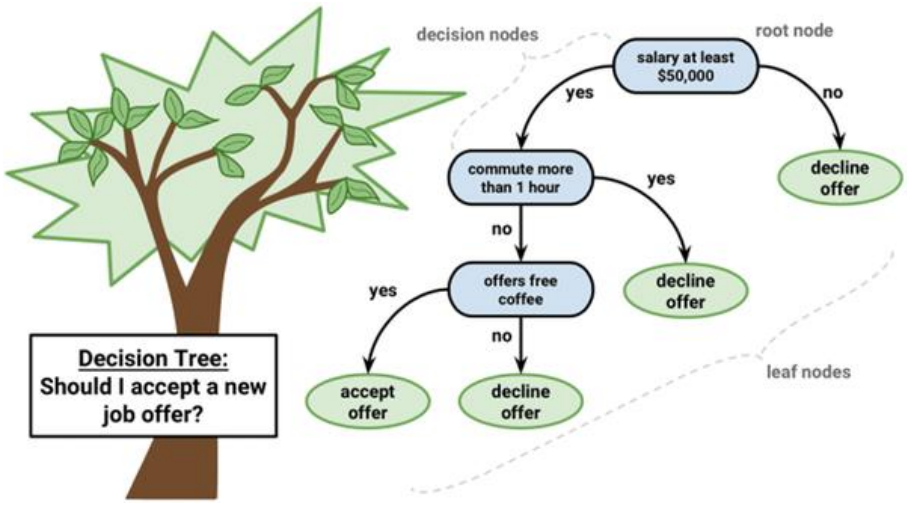

Figura 2 *Ejemplo de árbol de decisión*
Fuente: [4]

En el proceso de generación del árbol, se toma una estrategia «*greedy*», es decir, busca la solución subóptima en cada estado aspirando a que, la solución final, sea lo más cercana a la globalmente óptima. El hecho de tomar decisiones locales sin valorar de forma constante el rendimiento global otorga, entre otras ventajas, gran velocidad de cómputo. Además de esta sencillez en cuanto a cómputo se refiere, entre los motivos de su popularidad destacan también la

facilidad de comprensión, la posibilidad de utilizar datos que presenten colinealidad y la opción de emplear tanto valores discretos como continuos.

Sin embargo, presenta un problema importante de *over-fitting*, es decir, los árboles generados se adaptan demasiado bien a los datos introducidos y por tanto no son buenos generalizando. Es este el motivo por el que, finalmente, se utilizará el algoritmo de **Random Forest** en lugar de un árbol de decisión único.

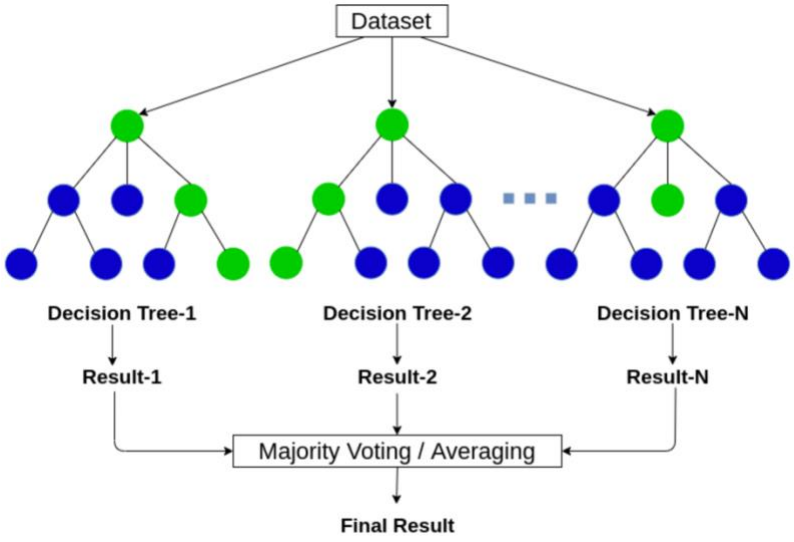

Figura 3 *Esquema del resultado del algoritmo Random Forest*
Fuente: [5]

Como se observa en la Figura 3, este modelo, que permite tareas tanto de clasificación como de regresión, basa su funcionamiento en la construcción de un conjunto de árboles de decisión durante el entrenamiento. Sin embargo, si todos los árboles se generaran de la misma forma y con los mismos datos, todos ellos serían iguales y el problema no se estaría solucionando. Para ello, y de ahí el nombre de Random, se sigue un proceso denominado «***Bagging***» que se compone de dos subprocesos y que se puede observar, a modo resumen, en la Figura 4:

- **Bootstrapping**: Para evitar que todos los árboles se entrenen con los mismos datos, este método de remuestreo permite formar subconjuntos de datos formados a partir de muestras aleatorias con repetición del conjunto de entrenamiento original. A su vez, es posible llevar a cabo, de forma simultánea, una «*feature selection*», es decir, no sólo utilizar un subconjunto de datos, sino que también emplear subconjuntos de las variables originales. Todo esto permite reducir el *over-fitting* que se produce en la utilización de un árbol de decisión único y la correlación entre árboles dentro del Random Forest. No obstante, es conveniente destacar que la subselección de muestras y atributos favorecerá la creación de árboles mediocres y otros especialmente buenos, pero el uso de un número relativamente grande de los mismos permite compensar estos efectos.

- **Aggregation**: La generación de múltiples árboles de decisión, genera también un problema, puesto que la variable de salida debe tomar un valor único. Para ello, se toman el conjunto de salidas y se realiza una votación por mayoría para determinar el output final. Existe también la opción del «*soft-voting*» para aquellos árboles cuyo

resultado sea una probabilidad y no un valor categórico, en los que se da más importancia a aquellos resultados muy seguros, es decir, cercanos a 0 o a 1.

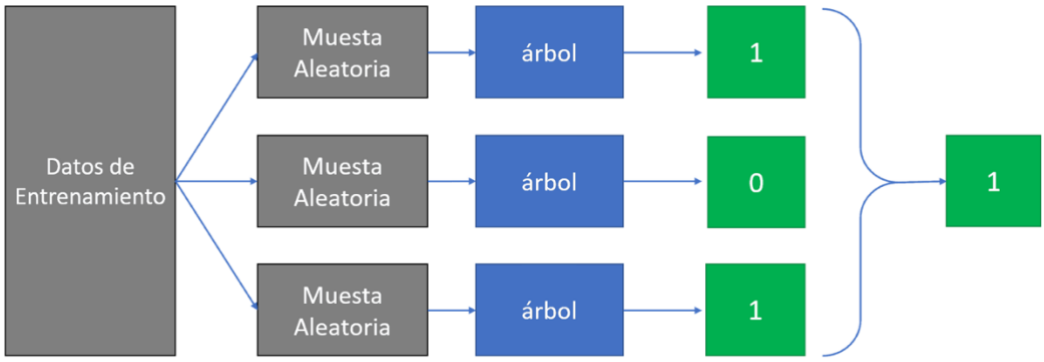

Figura 4 Proceso Bagging en el algortimo Random Forest
Fuente: [6]

De esta forma se consigue un modelo final altamente preciso, fácilmente interpretable y más dado a la generalización que utilizando un árbol de decisión único.

2.1.3 Redes Neuronales

Las redes neuronales se encuentran cada vez más presentes en el campo del Machine Learning puesto que, a pesar de la complejidad interna que presentan, existen numerosas librerías capaces de generar pequeñas y grandes redes neuronales con muy pocas líneas de código.

Para comprender el funcionamiento de las mismas, es necesario, en primer lugar, entender el funcionamiento de su elemento básico: Se entiende por **neurona artificial** una unidad de cálculo que intenta replicar el funcionamiento de las neuronas naturales del cerebro humano. Como vemos en la Figura 5, su comportamiento radica en la recepción de una serie de valores, aplicarle una serie de pesos que varían durante el entrenamiento, obtener un valor fruto de la suma de éstos y devolver el resultado.

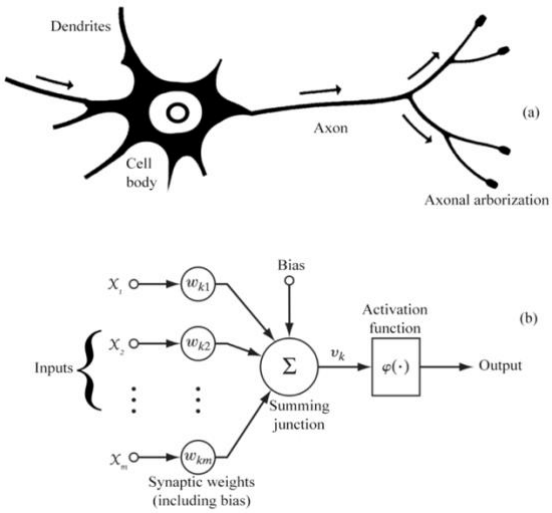

***Figura 5** Comparación de una neurona natural (a) con una artificial (b)*
Fuente: [7]

Con la intención de obtener relaciones no lineales entre las variables de entrada, se utilizan las **funciones de activación**, es decir, una transformación del valor obtenido mediante la combinación lineal de las entradas. Existen multitud de funciones distintas a emplear que permiten deformar la salida de forma no lineal, aunque a modo de ejemplo en la Figura 6 se puede observar la función Relu, una de las más utilizadas.

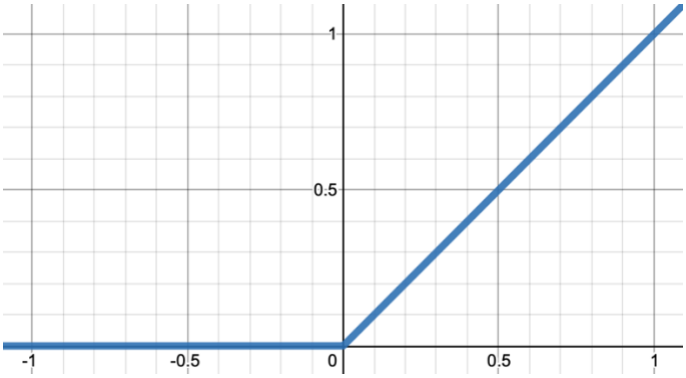

Figura 6 *Función Relu*
Fuente: Elaboración propia

Una vez entendido el funcionamiento de una neurona individual, ampliar al concepto de red neuronal es bastante sencillo. En primer lugar, utilizando un conjunto de una o más neuronas se puede generar una capa o Perceptrón, la forma más simple de red neuronal. A su vez, como se muestra en la Figura 7, conectando las salidas de una capa con las entradas de las adyacentes, es como se puede generar un modelo más complejo denominado **Perceptrón Multicapa** (MLP por sus siglas en inglés), compuesto por la capa de entrada, la de salida y un conjunto de capas ocultas o *hidden layers* cuyo número y complejidad pueden ser totalmente customizados [8].

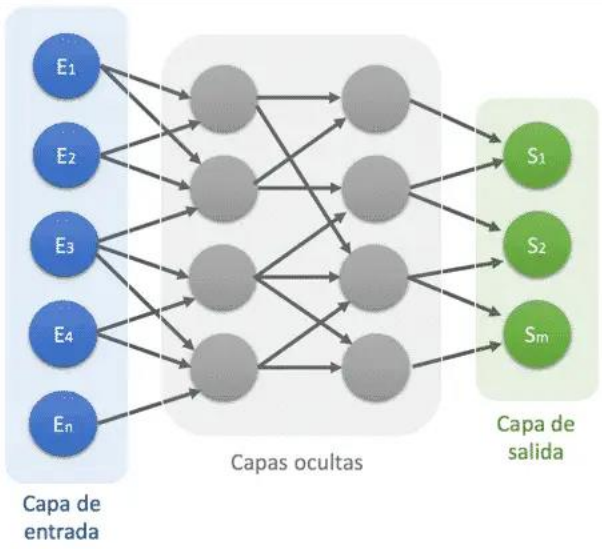

Figura 7 *Esquema red neuronal*
Fuente: [9]

Por último, es necesario abordar un punto importante que, si bien ya ha sido mencionado, debe ser explicado con más profundidad. Al explicar el concepto de neurona artificial y los pesos que utiliza, se ha comentado que éstos varían durante el proceso de entrenamiento, pero no se ha explicado cómo. En la inmensa mayoría de casos, el MLP utilizará un algoritmo denominado

«*backpropagation*», el cuál permite entrenar de forma eficiente la red neuronal. Para ello, se analiza desde la última capa de la misma la función de coste o error para, de forma iterativa y recursiva, modificar aquellos parámetros que generan mayor ruido en la clasificación o predicción de los datos, esperando que vaya convergiendo con la observación iterativa de muestras etiquetadas.

2.1.4 SVM

Las máquinas de vectores soporte (SVM por sus siglas en inglés) son un conjunto de algoritmos de aprendizaje supervisado que permiten tanto la clasificación como la regresión. Su funcionamiento radica en generar un hiperplano, es decir, un elemento divisor que permite separar el espacio multidimensional en dos mitades.

A simple vista, este comportamiento resulta similar al del MLP. Sin embargo, mientras que en el modelo anterior se busca un hiperplano capaz de separa los datos, en el caso del SVM, se busca separar de forma óptima el conjunto de entrenamiento. Es decir, se pretende encontrar aquel hipeplano que maximiza la separación entre clases. Gracias a ello, en la mayoría de los casos, sus resultados generalizan mejor y consiguen mejorar a los del Perceptrón Multicapa [10].

A modo de ejemplo, en la Figura 8 se puede observar un espacio bidimensional en el que el SVM generaría dicho hiperplano es decir, la recta que divide el conjunto de datos en dos maximizando la distancia de los elementos más cercanos de cada clase al vector soporte.

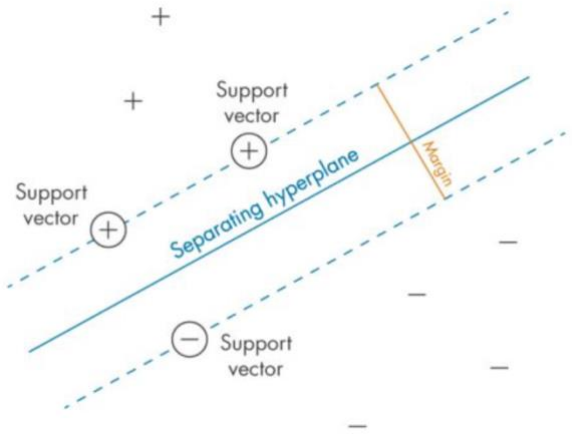

Figura 8 *Optimización de la distancia entre clases - SVM*
Fuente: [www.mathworks.com]

Debe ser considerado que no siempre los datos de entreno serán linealmente separables de forma absoluta y que, en caso de serlo, se podría estar dando un caso de *overfitting*. Con el objetivo de ofrecer cierta flexibilidad, se utiliza un parámetro de error que permite generar un margen de error bajo en la división.

Es interesante destacar que, para realizar la separación de los datos, la potencia de las máquiras de vectores soporte radica en el uso de funciones *kernel*, las cuales permiten aumentar las dimensiones de los datos. De esta forma, conjuntos de datos que, inicialmente, pueden no ser separables linealmente en n-dimensiones sí podrían serlo para n + m, como se muestra en la Figura 9. Existen multitud de funciones distintas, cada una con unas propiedades determinadas. Sin embargo, debido a que la mayoría de los problemas de clasificación de textos son linealmente

separables [11] y velocidad y eficiencia de usar un ***kernel* lineal**, decantará la elección de este tipo para el problema a resolver en este trabajo

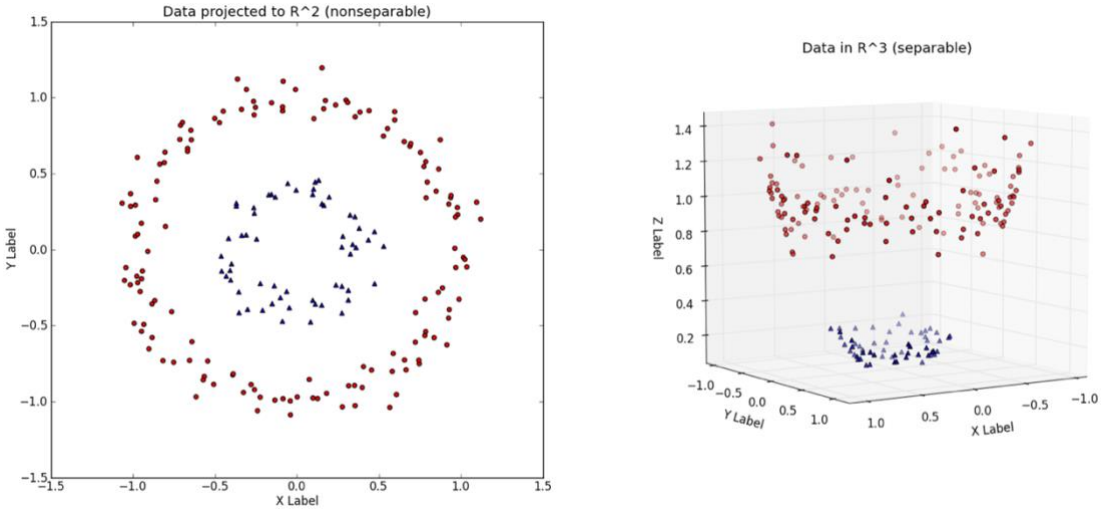

Figura 9 *Datos separables linealmente al pasar a tres dimensiones*
Fuente: [12]

2.1.5 Ensembles

Una técnica ampliamente utilizada gracias a las mejoras en la velocidad de cómputo y en la facilidad para obtener modelos es la creación de «***ensembles***», es decir, la generación de un conjunto de modelos de Machine Learning, cada uno con una predicción independiente de las del resto, cuyos resultados se combinan para obtener una predicción o clasificación única.

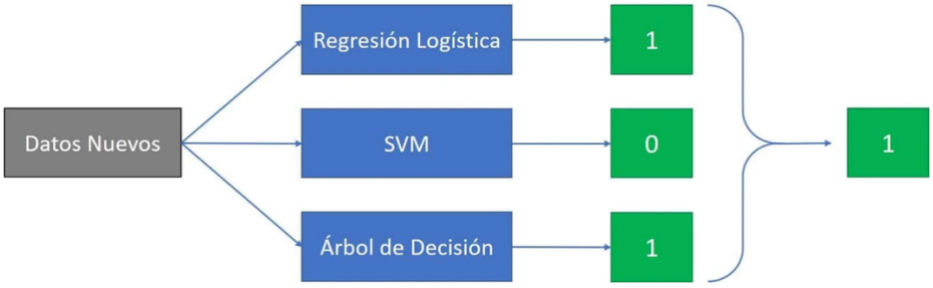

Figura 10 Ensemble de tres modelos de Machine Learning
Fuente: [13]

La principal ventaja que se obtiene de esto es que, al utilizar modelos diferentes con características y comportamientos distintos, sus errores tienden a compensarse, lo que redunda en una mejor generalización.

Una vez los modelos han sido creados, la decisión final puede ser tomada por voto de la mayoría, utilizando el *soft-voting*, o mediante la media aritmética, en función del tipo de datos utilizados.

2.2 Procesamiento del Lenguaje Natural

Dentro del ámbito de la inteligencia artificial la rama que estudia la interacción mediante el lenguaje humano entre las personas y los ordenadores recibe el nombre de **procesamiento del lenguaje natural** (**NLP** por sus siglas en inglés). Esta área de conocimiento a medio camino entre la informática y la lingüística se encarga de diseñar y definir mecanismos eficaces para la simulación, comprensión e interpretación del lenguaje natural por parte de los ordenadores.

2.2.1 Análisis del Lenguaje Natural

Cuando se habla del análisis del lenguaje natural es importante realizar una clasificación en cuatro categorías distintas, las cuales se exponen a continuación:

- **Análisis morfológico**: Se encarga de realizar el análisis de las palabras para extraer la raíz, las conjugaciones, los rasgos flexivos y otros aspectos del lenguaje que se utilizan en la formación de palabras.

- **Análisis sintáctico**: Su función es la de estudiar y entender los principios y las reglas del lenguaje que determinan la combinación posible entre palabras para la formación de unidades superiores como son las oraciones.

- **Análisis semántico**: Cuando se habla de semántica se hace referencia al significado tanto de palabras como de oraciones. Por tanto, este aspecto de la lengua trata de estudiar el mensaje que se quiere transmitir, así como las dualidades propias del lenguaje humano.

- **Análisis pragmático**: Mediante este análisis se pretende no sólo discernir el significado de las palabras y oraciones como tal, sino que busca comprender el mensaje transmitido dentro de un contexto más amplio. Por tanto, esta rama de estudio no sólo trata el lenguaje en sí, sino que debe entender la cultura, la situación psicolingüística y las relaciones interpersonales entre los sujetos de la comunicación entre otros factores.

Figura 11 *Facetas del análisis del lenguaje*
Fuente: Elaboración propia

De la lista anterior se puede inferir una relación jerárquica y acumulativa en la que cada uno de los niveles implica alcanzar un grado de conocimiento del lenguaje superior al anterior, como se muestra en la Figura 11. Como es de suponer, el estudio del lenguaje en profundidad es una disciplina compleja y tediosa por lo que, para el presente trabajo, la palabra será la unidad única del lenguaje utilizada y, por tanto, se realizará única y exclusivamente un análisis a nivel morfológico y semántico, obviando la existencia de unidades del lenguaje superiores.

2.2.2 Fases del procesado de palabras

El tratamiento de las palabras, empero, requiere de un preprocesado previo al uso de las mismas por parte del ordenador. Para ello, en el presente proyecto se ha seguido un proceso de cinco fases que se explicarán a continuación:

- **Minúsculas**: La codificación interna en un ordenador de los caracteres se basa en los códigos ASCII, los cuales carecen de significado intrínseco y se utilizan, única y exclusivamente, a modo de representación. Por ello, mayúsculas y minúsculas cuentan con valores diferentes lo cual impide que el ordenador sepa que se está haciendo referencia a lo mismo. De esta forma, el primer paso será convertir todo el texto a minúsculas.

- **Eliminar caracteres**: Del mismo modo que las letras se representan mediante códigos ASCII, lo mismo ocurre para otros caracteres como números o incluso puntos, comas, etcétera. Por este motivo, el siguiente paso será eliminar todos aquellos caracteres que resultan innecesarios y que no aportan información alguna. En el caso de este trabajo, en la Figura 12 se pueden observar los caracteres que, junto a los números, serán eliminados.

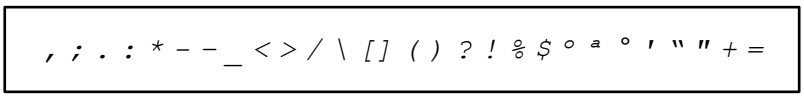

Figura 12 *Conjunto de caracteres a eliminar*
Fuente: Elaboración propia

- **Tokenización**: Hasta este momento, los textos se han tratado como una unidad compuesta por distintos caracteres sobre los que realizar las acciones anteriores descritas. Sin embargo, como ya se ha comentado, la unidad sobre la que se va a realizar el análisis es la palabra, por lo que es necesario hacer una división del texto en un conjunto de éstas. Este proceso se denomina tokenización y consistirá en obtener una lista de palabras, o *tokens*, utilizando, en este caso, el carácter de espacio como separador.

- **Stopwords**: Una vez ya se ha conseguido un listado de palabras, conviene realizar una reflexión acerca de la utilidad de las mismas. A modo de ejemplo, usaremos la frase «La casa azul.». Transformando este texto en una lista de palabras mediante los pasos anteriores, el resultado obtenido sería «[‘la’, ‘casa’, ‘azul’]». No obstante, es evidente que no todas las palabras aportan el mismo significado, puesto que «casa» es indispensable para saber de qué se habla, «azul» aporta información estrictamente relevante, pero «la» es un determinante que, aunque a nivel sintáctico y de comunicación es necesario, a nivel semántico se convierte en totalmente

prescindible. Por este motivo, utilizar palabras como preposiciones, artículos o determinantes no sólo no aporta valor, sino que entorpecería el análisis posterior. Este conjunto de palabras que cumplen dicho criterio se denomina *stopwords* y deben ser eliminadas de la fase anterior.

- **Stemming/Lemmatization**: El último paso de estas cinco fases se corresponde con el llamado análisis morfológico. El lenguaje humano es complejo a la par de extenso, prueba de ello es que, en muchos casos, un conjunto amplio de palabras con pocas letras de diferencia, pueden tener prácticamente el mismo significado. A modo de ejemplo podemos utilizar la conjugación de un verbo, como puede ser «escribir», en el que el conjunto de palabras «escribo», «escribe» y «escribes» hacen todas referencia al acto de escribir. Aunque una persona puede ver fácilmente la conexión entre estas palabras hay que recordar que el ordenador solamente ve códigos numéricos, por lo que una opción para introducir estas relaciones es devolverlas todas ellas a su raíz morfológica. Para ello existen principalmente dos procesos a seguir: El primero de ellos, el «*stemming*», consiste en truncar las palabras a un valor común, en este caso «escrib» mientras que el segundo, el «***Lemmatization***», busca la raíz verdadera de las palabras, en este caso el verbo «escribir». Aunque el segundo proceso es más complejo suele ser más adecuado y preciso por casos como el del verbo «ser», en el que las palabras «soy» y «es» tienen la misma raíz pero ningún patrón de caracteres común.

Figura 13 *Cinco fases del preprocesado de palabras*
Fuente: Elaboración propia

Una vez se han completado las cinco fases anteriormente explicadas, el texto inicial se habrá convertido en un conjunto de palabras filtradas y preparadas para su posterior uso en el análisis del lenguaje pertinente.

2.2.3 Matriz TF-IDF

Como se ha venido mencionando, las palabras obtenidas del proceso anterior se encuentran representadas en formato textual. Aunque esto sea útil para las personas, para el ordenador recibir una serie de caracteres con una codificación arbitraria no es relevante. Por este motivo, se deben transformar los datos a tipo numérico.

Con dicho objetivo, existen numerosos métodos a la hora de tratar con palabras, sin embargo, para el presente estudio, se ha tomado la decisión de utilizar la popular y extendida matriz **TF-IDF**. El proceso de transformación de palabras a la citada matriz utiliza un peso calculado en base a una medida estadística que permite evaluar la. importancia de una palabra respecto a un documento y en relación al conjunto de ellos. Para mejorar la comprensión de dicha matriz se explicará, a continuación, el cálculo del mencionado peso, el cual consta de dos valores:

- **TF**: El término TF hace referencia al concepto de *Term Frequency*, es decir, para cada uno de los textos se extrae la lista de todos los términos únicos que aparecen en él, se calcula la frecuencia relativa de aparición y se divide entre el número total de palabras del texto. De esta forma se da un valor superior a aquellos términos que aparecen de forma más frecuente en cada documento.

- **IDF**: El segundo término que da nombre a la matriz hace referencia al concepto de *Inverse Document Frequency*, el cual calcula la importancia de un término respecto al conjunto de textos. Dicho de otra forma, lo que este valor hace es calcular la inversa del número de textos en los que aparece el término en cuestión entre el número total de textos. De este modo el segundo término prima los términos que aparecen en pocos documentos.

$$W_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$W_{i,j}$ = peso de la palabra i para el texto j

$tf_{i,j}$ = frecuencia relativa del término i en el texto j

df_i = número de textos en los que aparece el término i

N = número total de textos

Figura 14 *Cálculo de los pesos de la matriz TF-IDF*
Fuente: Elaboración propia

De esta forma, se obtiene un peso, es decir, un valor numérico para cada uno de los términos en cada uno de los textos que, posteriormente, serán los datos de entrada para los modelos a utilizar. A modo de ejemplo, en la Figura 15 podemos observar como una palabra que aparece en todos los textos dados carece de valor.

j1: "La casa azul." $W_{«casa»,j1} = 1/3 \times \log\left(\frac{2}{2}\right) = 0$

j2: "La casa roja." $W_{«casa»,j2} = 1/3 \times \log\left(\frac{2}{2}\right) = 0$

Figura 15 *Ejemplo del cálculo de la matriz TF-IDF*
Fuente: Elaboración propia

2.3 Exploración espacial

Tras la finalización de la Segunda Guerra Mundial y en plena Guerra Fría, las dos potencias mundiales resultantes del conflicto (Estados Unidos y la Unión Soviética) afrontaron otra batalla que, si bien no tenía carácter bélico, definiría los comienzos del viaje de la humanidad más allá de la Tierra. Durante la llamada Carrera Espacial [14], ambos países se enfrentaron por dominar la exploración espacial a través del uso de satélites artificiales y por ser los primeros en pisar la Luna.

Hoy, cincuenta años después de ese momento, se vive de nuevo una carrera espacial que, aunque totalmente distinta a la anterior, vuelve a ser todo un reto tanto tecnológico como

transcendental en el desarrollo de la especie humana. Tanto el contexto político como social de ambos momentos son radicalmente distintos, sin embargo, conviene destacar aquellos dos factores que se considera clave para comprender la coyuntura actual.

En primer lugar, si antaño los protagonistas fueron los Estados, en la actualidad, las empresas privadas han tomado el papel de actores principales de la futura exploración espacial. Aunque se podría hablar de varias de ellas, estudiaremos brevemente las tres que han logrado los mayores avances. Por un lado, **Virgin Galactic**, empresa fundada por el magnate de los negocios Sir Richard Branson, se ha convertido en la primera entidad privada en realizar un viaje turístico a más de ochenta kilómetros de altura en julio de 2021. Gracias a ello, es considerada una de las principales promesas de cara al futuro turismo espacial, aunque por el momento su tecnología se encuentra limitada a vuelos suborbitales.

Por otro lado, **SpaceX**, nacida en 2002 de la mano del conocido emprendedor Elon Musk, es actualmente conocida a nivel mundial por construir el primer cohete de financiación privada en alcanzar la órbita terrestre, ser los primeros en lanzar y aterrizar con éxito un cohete reutilizable o ser la primera empresa privada en poner a seres humanos en órbita. Todo ello la ha situado como un referente para las futuras expediciones espaciales, llegando a ser recientemente ganadora de un contrato valorado en 2.900 millones de dólares de la NASA [15] para llevar a humanos de nuevo a la superficie lunar bajo el programa Artemis, del que hablaremos más adelante. Además, la empresa cuenta con un programa propio con el que espera llevar a lo primeros humanos al planeta rojo en 2024 y «convertir a la humanidad en una especie multiplanetaria»[2].

Por último, **Blue Origin** fundada por Jeff Bezos en el año 2000, aunque más discreta que las anteriores no se queda atrás en la carrera espacial y en el mismo mes de julio subastó el primer billete turista para viajar al espacio en su propio cohete reutilizable, alcanzando la misma un valor de veintiocho millones de dólares.

En segundo lugar, el continuo avance científico que se ha experimentado a lo largo de las últimas décadas ha permitido una gran mejora en la tecnología de los viajes espaciales. Además, si junto a este último hecho se añade la aparición de empresas privadas y la competencia entre ellas, se consigue explicar en gran medida la notable reducción de costes que vemos en la Figura 16.

[2] Web oficial: https://www.spacex.com/human-spaceflight/mars/

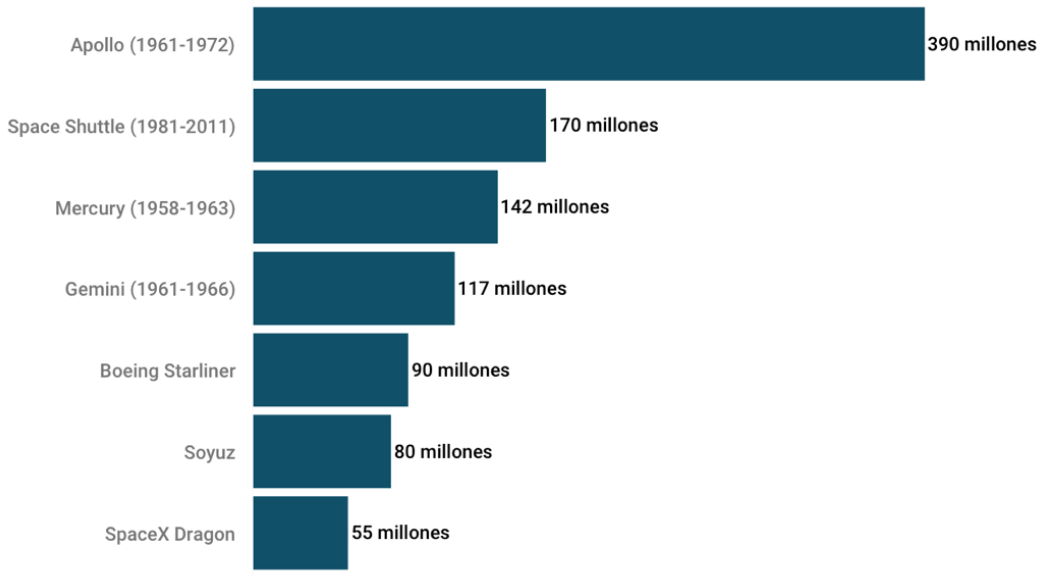

Figura 16 *Coste estimado por asiento en cohetes espaciales ajustado por inflación*
Fuente: Statista y NASA

De esta forma, gracias al mencionado progreso tecnológico, la mejora en la eficiencia económica, así como al periodo de paz y cooperación en el que vive la humanidad, las metas a plantear son más ambiciosas que nunca y el interés cada vez mayor. Por ello, para los futuros viajes espaciales ya no solo se plantea como destino la Luna, sino que la llegada del primer ser humano a Marte se presenta ya como una realidad próxima. Prueba de ello es el programa Artemis, un proyecto conjunto de la Agencia Espacial Europea (ESA), la Canadiense (CSA), la Australiana (ASA), la Japonesa (JAXA), compañías privadas y, como actor principal, la NASA. Con él se pretende llevar a la humanidad de nuevo a la Luna y establecer las bases de la futura llegada de personas a Marte[3], dando un salto cualitativo y entrando de lleno en la exploración espacial al sobrepasar la órbita lunar.

A su vez, cabe destacar que la salida de la Tierra puede suponer un salto cualitativo en la investigación científica. En un informe de la NASA sobre la exploración humana de Marte [16] ya se indican numerosos objetivos y preguntas que se plantean responder con la llegada a Marte: desde una mejor compresión del clima, hasta saber si existe vida fuera de nuestro planeta pasando por una mejor comprensión de su composición y de su pasado. Además, no es necesario llegar a la superficie de otro planeta para poder desarrollar avances científicos, sino que la simple salida de la Tierra supone una gran oportunidad. Por ejemplo, experimentos en condiciones de gravedad cero o el uso de telescopios sin la interacción de la atmósfera terrestre ya suponen actividades imposibles de llevar a cabo en nuestro planeta y, por tanto, con enorme interés científico.

2.4 Inteligencia Artificial y Sistemas médicos en el espacio

La tecnología y la medicina han recorrido desde siempre un camino en paralelo. Los avances tecnológicos han repercutido de forma decisiva en la mejora tanto de la propia salud de las personas como en las herramientas utilizadas para conservarla. Es un hecho el uso de la **robótica**

[3] Web oficial del programa Artemis: https://www.nasa.gov/specials/artemis/

en el campo de la medicina para, por ejemplo, la mejora en la maquinaria para el diagnóstico o el uso cada vez más frecuente de exoesqueletos para pacientes con movilidad reducida [17].

De la misma forma, el uso de la **AI** será cada vez más común dentro de los procesos médicos, llegando a ser una herramienta fundamental y cotidiana dentro del sistema sanitario. Ejemplo de ello es el uso de redes neuronales profundas en la clasificación de tumores cerebrales, existiendo en la actualidad experimentos que demuestran que su uso mejora la precisión en la tipificación de los mismos de forma notoria [18].

La utilización de **sistemas informáticos para la ayuda en la decisión clínica** (**CDSS** por sus siglas en inglés) en hospitales para facilitar el trabajo a los profesionales sanitarios es un hecho también desde hace años [6] por lo que implementar un soporte de ayuda a la decisión de estos profesionales podría suponer tanto una ayuda para ellos como una mejora en la calidad de las decisiones tomadas.

Por otro lado, la necesidad de crear modelos de aprendizaje a partir de datos reales podría verse desde dos perspectivas distintas: por un lado, un modelo capaz de ayudar en la toma de decisiones de las personas gracias a la generación de predicciones a partir de datos médicos históricos y por otro la utilización de estos datos y modelos para generar una suerte de paciente a medida.

2.4.1 Virtual patients

En muchas ocasiones, la inexistencia de datos o pacientes reales puede suponer un obstáculo en la investigación médica. Por este motivo, es especialmente interesante la creación de pacientes diseñados como avatares digitales capaces de replicar casos clínicos específicos. A este último concepto se le conoce, en el mundo de la investigación, como ***virtual patient*** y sus potenciales beneficios en el aprendizaje del personal sanitario ya están siendo comprobados [19].

El proceso de creación de estos pacientes virtuales se puede concretar en cuatro fases: establecer los requerimientos, diseñar los pacientes, realizar el prototipado y llevar a cabo evaluaciones [20]. Con ello, se puede establecer un marco para la creación indiscriminada de casos clínicos que permita al personal médico estudiar y valorar numerosas situaciones que podrían llegar a darse en la realidad.

Además, en el caso del presente proyecto en el que es necesaria una gran base de datos para entrenar los modelos de ML, esto puede suponer un salto cuantitativo en el resultado final obtenido, ya que, en la mayoría de los casos de análisis de datos, la muestra utilizada supone el principal requisito para el éxito o el fracaso.

2.4.2 CIE-9-MC

El acrónimo **CIE**[5] se corresponde con la **clasificación internacional de enfermedades** elaborada por la Organización Mundial de la Salud cuya edición actual se remonta a la de los años 90 (aunque ya existe una nueva versión preparada para entrar en vigor en 2022). Su función radica en generar una codificación universal para las distintas afecciones y categorías que pueden sufrir las personas a nivel médico de forma que el dictamen de la patología se vuelve agnóstico en

[5] Edición empleada: *https://eciemaps.mscbs.gob.es/ecieMaps/browser/index_9_mc.html*

cuanto al idioma. De esta forma estos códigos resultan especialmente útiles en colaboraciones internacionales y en labores de traducción médica.

En el caso del presente trabajo, se utilizará en concreto la versión CIE-9-MC tanto en las patologías a estudiar como en la base de datos de casos clínicos de la que dispondremos para el proyecto, como veremos en el Capítulo 3.

Por último, debemos destacar la lógica detrás de la codificación en la edición utilizada. Cada código presenta una estructura del tipo CCC.SP, siendo el número CCC el que define la categoría de la enfermedad, S la subcatergoría y P la subclasificación. A su vez, se utilizan distintas letras como la V o la E para clasificar condiciones distintas como, por ejemplo, factores de riesgo.

2.5 MEDEA

Como ha sido mencionado previamente la exploración espacial pretende dar un salto cualitativo en la década de 2030. No obstante, a pesar de la relativa cercanía en el tiempo a esta fecha, todavía existen muchos obstáculos que superar entre los que habría que destacar, por la temática de este trabajo, el de la salud de los astronautas. El aislamiento, las distintas fuerzas gravitatorias, la radiación, el aumento considerable en la duración de las misiones espaciales o el verse obligados a vivir en hábitats hostiles hace del cuidado de la salud de los astronautas todo un desafío.

Además, los motivos planteados en la motivación (Apartado 1.1 de este documento), hacen que los actuales sistemas de cuidado médico, basados en última instancia en la comunicación con la Tierra, sean inviables para estos nuevos escenarios. Por ello, diseñar un sistema autónomo de ayuda a la decisión clínica con soporte en tiempo real se presenta como pieza angular en esta nueva etapa de la humanidad.

En el caso de este proyecto, dicho sistema se denominará **MEDEA** [21], un sistema de asistencia médica que deberá cumplir los siete criterios de tiempo real, autonomía, adaptabilidad, variables de entrada consciencia del entorno optimización y estándares que ya han sido desarrollados anteriormente. Para lograrlo, será necesario utilizar tecnologías de Machine Learning, análisis de datos, procesamiento del lenguaje natural y modelos clasificadores, como redes neuronales, árboles de decisión o clasificadores lineales, entre otros.

Asimismo, el diseño de este sistema debe tener en cuenta que su cometido es ayudar a tomar decisiones en el espacio, por lo que debe estar centrado en las patologías que se pueden desarrollar con mas probabilidad fuera de nuestro planeta. Para ello se utilizarán estudios en los que ya han sido identificados riesgos asociados a viajes espaciales de larga duración [22] o las condiciones médicas más probables en estos casos, siendo el contenido de este último trabajo un elemento clave en el desarrollo de este estudio [1].

Como se puede observar en la Figura 17, para lograr este sistema se han planteado cuatro módulos interdependientes capaces de obtener en su conjunto la funcionalidad deseada:

- **CDSS autónomo y en tiempo real**: Este primer módulo será el encargado de realizar las tareas de clasificación y predicción pertinentes dentro del sistema. Para ello contará con tres submódulos especializados en tareas concretas: El primero de ellos, mediante modelos de Machine Learning como el MLP, tendrá la función de tomar decisiones acerca del riesgo vital, la demorabilidad o incluso los diagnósticos compatibles con la situación a analizar. El segundo se encargará de distinguir entre

las posibles intervenciones a realizar en base a lo obtenido por el submódulo anterior. Y, por último, el tercero de ellos se dedicará a definir el plan de acción óptimo para el contexto de la situación. Como ya ha sido comentado, el espacio exterior será un entorno adverso en el que la escasez de recursos o la necesidad de conservación de maquinarias, habitáculos, medios de transporte, etcétera, hará que prime el tomar decisiones que, en circunstancias «normales», serían distintas. Por ello, los planes de acción deben de optimizar al máximo posible todas las variables mencionadas incluida, por supuesto, la salud del paciente en cuestión. De esta forma, el módulo en su totalidad sopesará la situación y ofrecerá el conjunto de acciones óptimas a llevar a cabo.

- **Aprendizaje espacial adaptativo**: Si bien el módulo anterior será el encargado de clasificar y tomar las decisiones pertinentes, para llegar a ese punto será necesario haber entrenado los modelos de Machine Learning previamente a partir del conjunto de datos disponibles para dicha tarea. No obstante, este proceso no es único, sino que se deberá realizar de forma iterativa a lo largo del tiempo puesto que, en todo momento, se irán incorporando nuevos datos. Además, será ineludible realizar un análisis y procesado exhaustivo de los mismos para garantizar la calidad de los modelos resultantes de forma constante [23]. Por último, cabe mencionar que este proceso será el único a realizar en instalaciones en la Tierra, donde se puede emplear una mayor capacidad de cómputo.

- **Interoperabilidad semántica:** Este módulo será el encargado de definir los procesos y los sistemas de transmisión de los datos, así como del intercambio de los mismos entre los distintos módulos del sistema. Además, en él se definirá el uso de conceptos como los CIE o los SNOMED Clinical Terms.

- **Soporte ético y legal:** El cumplimiento con la regulación de la Protección de Datos y la privacidad tanto de los usuarios del propio sistema como de los sujetos cuyos datos han sido utilizados en el proceso de aprendizaje debe ser una prioridad. Sin embargo, ello deberá ir acompañado de la minimización de pérdida de información. Este último módulo será el encargado de buscar el equilibrio y garantizar el funcionamiento del sistema respetando la confidencialidad y la aplicación de los más altos estándares de calidad.

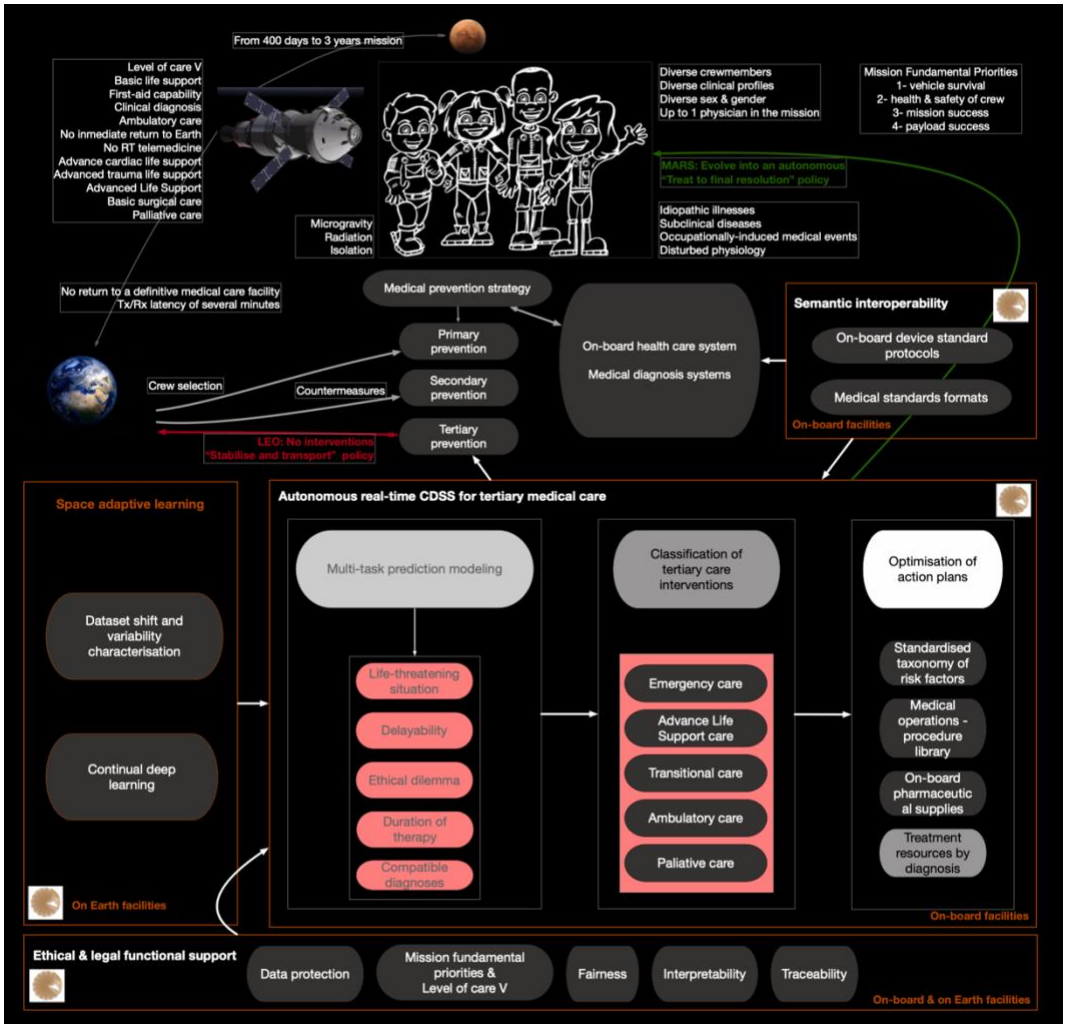

Figura 17 MEDEA y sus cuatro módulos
Fuente: [21]

Con todo ello se ha definido un sistema capaz de solucionar un problema vital de cara a la futura exploración espacial, una actividad con un enorme potencial humano, científico y económico en la que el desarrollo de un sistema como el descrito anteriormente podría ser decisivo.

Sin embargo, este sistema es realmente complejo, por lo que en el presente proyecto se realizará una aproximación al módulo del CDSS autónomo y, en concreto, al primer submódulo encargado de llegar a cabo las tareas de predicción en cuanto al riesgo vital y la demorabilidad. Para ello, nos centraremos en las cien patologías más comunes en el espacio [1] partiendo de una base de datos de casos clínicos del 112 cedida por la Generalitat Valenciana y, para suplir sus limitaciones, se empleará literatura médica especializada y revisada por un profesional médico.

3 Materiales

En este capítulo se analizará el conjunto de elementos que han sido necesarios para la realización de los experimentos asociados a este proyecto, desde los datos utilizados hasta el equipamiento hardware y software empleado.

3.1 Conjunto de datos empleados

Para la realización del presente trabajo han sido utilizados datos provenientes de distintas fuentes de información claramente diferenciadas. En concreto, una base de datos del centro de emergencias (teléfono 112) de la Comunidad Valenciana, un portal de internet para la consulta de información médica por parte de médicos e investigadores y fuentes de información complementarias.

El motivo de usar distintas fuentes de información viene dado por la voluntad de acometer un clasificador especializado **en el espacio**. Se debe tener en cuenta que las patologías más frecuentes en dicho entorno no serán las mismas que en la Tierra, por lo que ha sido necesario obtener información alternativa sobre aquellas enfermedades que resultarán más probables en la exploración espacial y de las que disponemos de pocos casos reales. Para ello, como vemos en la Figura 18 nos hemos basado en las condiciones médicas referenciadas en el artículo de los investigadores Elkin Romero y David Francisco sobre la mitigación de riesgos en la exploración espacial [1].

1	Abdominal Injury	26	Burns (secondary to fire)	51	Hemorrhoids	77	Respiratory Infection
2	Abdominal Wall Hernia	27	Cardiogenic Shock (secondary to infarction)	52	Herpes Zoster	78	Retinal Detachment
3	Abnormal Uterine Bleeding	28	Chest Injury	53	Hip Sprain/Strain	79	Seizures
4	Acute Arthritis	29	Choking/Obstructed Airway	54	Hip/Proximal Femur Fracture	80	Sepsis
5	Acute Cholecystitis/Biliary Colic	30	CO_2 Headache/ICP/Cognitive	55	Hypertension	81	Shoulder Dislocation
6	Acute Compartment Syndrome	31	Constipation (SAS)	56	Immune System Dysfunction/Illness	82	Shoulder Sprain/Strain (EVA)
7	Acute Diverticulitis	32	Decompression Sickness (secondary to EVA)	57	Indigestion	83	Skin Abrasion/Laceration
8	Acute Glaucoma	33	Dental: Exposed Pulp, Caries, Abscess, Tooth Filling/ Crown Loss	58	Influenza	84	Skin Infection
9	Acute Pancreatitis	34	Depression	59	Insomnia	85	Skin Rash
10	Acute Prostatitis	35	Diarrhea	60	Knee Sprain/Strain	86	Small Bowel Obstruction
11	Acute Radiation Syndrome	36	Dust Exposure (celestial)	61	Landing Loads/Injuries	87	moke Inhalation
12	Acute Sinusitis	37	Elbow Dislocation	62	Lower Extremity Stress Fracture	88	Space Adaptation Sickness (SAS)/Neurovestibular
13	Aerobic Capacity Loss	38	Elbow Sprain/Strain	63	Lumbar Spine Fracture	89	Stroke
14	Allergic Reaction (mild to moderate)	39	Electric Shock Injury	64	Malnutrition	90	Sunlight Exposure/Sunburn
15	Altitude Sickness/Hypoxia	40	Eye Abrasion (foreign body)	65	Microbial-Host Interaction	91	Sudden Cardiac Arrest
16	Angina/Myocardial Infarction	41	Eye Chemical Burn	66	Medication Overdose/Reaction	92	Toxic Exposure (ammonia, etc.)
17	Anaphylaxis	42	Eye Corneal Ulcer	67	Mouth Ulcer	93	Traumatic Hypovolemic Shock
18	Ankle Sprain/Strain	43	Eye Infection	68	Muscle Atrophy	94	Urinary Incontinence (SAS)
19	Anxiety	44	Eye Penetration (foreign body)	69	Nasal Congestion (SAS)	95	Urinary Retention (SAS)
20	Appendicitis	45	Finger Dislocation	70	Nephrolithiasis (renal stone)	96	Urinary Tract Infection
21	Atrial Fibrillation/Flutter	46	Fingernail Delamination (EVA)	71	Neurogenic Shock	97	Vaginal Yeast Infection
22	Back Injury	47	Gastroenteritis	72	Nose Bleed (SAS)	98	SANS/VIIP – Spaceflight Associated Neuro-Ocular Syndrome Increased Intracranial Pressure
23	Back Pain (SAS)	48	Head Injury	73	Orthostatic Intolerance	99	Wrist Fracture
24	Barotrauma (sinus block)	49	Headache (late SAS)	74	Otitis Media/Externa	100	Wrist Strain/Sprain
25	Behavioral Emergency	50	Hearing Loss	75	Paresthesia		
				76	Pharyngitis		

Figura 18 *100 condiciones médica más probables en misiones de exploración espacia*
Fuente: [1]

3.1.1 Base de datos de pacientes reales

Al inicio de la realización de este trabajo, el departamento de la Conselleria de Sanitat Universal i Salut Pública y la Agencia Valenciana de Seguridad y Respuesta a las Emergencias nos concedió acceso a una base de datos propia en la que se almacenan las llamadas efectuadas al 112 dentro de la Comunidad Valenciana.

Esta base de datos consta de un total de 136.160 casos clínicos representativos, haciendo referencia. cada uno de ellos, al paciente objeto de la llamada a emergencias y 193 columnas en los que se almacena la información del mismo. A su vez, para describir la base de datos, podemos dividir estos atributos en seis subgrupos principales:

- **CIE**: El código CIE-9-CM, explicado en el Punto 2.4.2, asignado al paciente es varios momentos temporales como el asignado durante la llamada, el valorado por el facultativo a la llegada al hospital en caso de ocurrir y el código finalmente asignado tras la oportuna evaluación médica. Tras un simple vistazo, observamos que muchas de estas celdas se encuentran vacías, pues no todos los pacientes siguen el mismo procedimiento (algunos llegan a acudir al hospital mientras en otros casos no es así) por lo que, para nuestro estudio, hemos decidido utilizar la columna denominada «***DIA_CODIGO***» al ser esta la que almacena el último código registrado.

- **VARIABLES**: Debido al uso de esta base de datos en otros proyectos ajenos al presente trabajo, los datos del estado del paciente se encuentran almacenados en base a unas sintomatologías concretas en las que se almacena. Para ello, existe un conjunto de columnas en las que se realiza una anotación en texto. A modo de ejemplo, la variable *«VAR_Alteración de la conducta»* presenta valores como « *AGITACION/ AGRESIVIDAD*». Para discernirlas del resto, estas variables presentan al inicio del nombre el prefijo «***VAR***» y se le suma la columna «*ARB_VARIABLES*», lo que hace un total de ochenta columnas de este tipo.

- **FLAGS**: Existe un conjunto de atributos que señalan la presencia o no de determinados elementos en los casos clínicos almacenando un valor numérico. En particular, podemos destacar los que indican la presencia o no de código CIE a la llegada al hospital, el subgrupo de edad al que pertenece el paciente o si el caso clínico presenta, a ojos del operador, riesgo o no para el paciente y si la emergencia puede ser demorada en el tiempo. De todos ellos, utilizaremos el atributo «***LAB_RIESGO_VITAL***» como etiqueta para indicar si el caso se puede considerar como de riesgo vital, cuya codificación se basa en un dos para denotar la presencia de riesgo y de un uno para el caso contrario.

- **OBSERVACIONES**: Esta variable, obtenida de la columna «***INC_OBSERVACIONES***» recopila el texto libre introducido durante la comunicación con el 112 por operador como cadena de caracteres. Un ejemplo podría ser «tos ahogos desde anoche con eferalgan remite piden ambulancia» en el que se puede observar que se realiza una breve descripción de los síntomas y la descripción de la situación general.

- **DATOS DEL PACIENTE**: Los datos como la edad, el sexo o la ubicación del paciente se almacenan en distintas columnas para tal efecto. Cabe destacar que la

edad se almacena en días en el atributo «***CALC_AGE***» y el sexo codificado como *M* para mujer y *H* para hombre en la variable «***PAC_SEXO***».

De esta forma, la base de datos inicial sufrirá una primera transformación para cumplir con los requerimientos de este proyecto mediante el filtrado de datos, quedando una nueva tabla de 136.160 filas y cuatro columnas en total, como podemos ver en la Tabla 1.

CIE	Lista con el/los CIEs asignados en última instancia.
SEX	Sexo del paciente: H para hombre y M para mujer.
AGE	Edad del paciente en días naturales
RISK	Valor de 1 si el caso NO presenta riesgo y 2 en caso contrario.
TEXT	Concatenación del texto de todas las variables del caso junto al texto libre introducido por el operador.

Tabla 1 *Primer filtrado de la Base de datos del 112*
Fuente: Elaboración propia

3.1.2 Literatura médica

Como ya ha sido comentado anteriormente, uno de los objetivos de este proyecto en la realización de pacientes virtuales coherentes con determinadas patologías asociadas a misiones de exploración espacial [1]. Para ello, es imprescindible contar con una fuente de información amplia, fiable, específica y, además, digitalizada.

Tras la consulta de diversos portales web que encajaban con la descripción anterior, la decisión final quedó en dos fuentes de conocimiento médico: por una parte, el proveedor de información online especializado en medicina **UpToDate**[6] y por otra, la base de datos de conocimientos médicos **eMedicine**[7]. Ambos portales abarcan prácticamente todos los campos médicos referenciando artículos científicos y ofreciendo una síntesis de la información clasificada y organizada por enfermedades, entre otros criterios, permitiendo, por tanto, una búsqueda rápida y sencilla.

Aunque a simple vista parecen servicios similares, existen tres puntos clave que nos permitieron decantarnos por uno de ellos:

- **UpToDate** presenta, por término medio, más información en cuanto a los procedimientos a llevar a cabo y la medicación de cada, aunque no es una norma en la totalidad de los casos y en la inmensa mayoría ambas plataformas ofrecen documentación más que suficiente.

[6] Web oficial: https://www.uptodate.com/contents/search
[7] Web oficial: https://emedicine.medscape.com

- **eMedicine** presenta información más estructurada. De forma general, mientras que UpToDate no sigue ningún patrón a nivel general, en eMedicine encontraremos un esquema con un punto llamado «*Presentation*» que incluirá, entre otras, una página de «*History*» en la que se revisan los aspectos más funamentales de la patología y otra de «*Physical Examination*» en la que se ofrecen indicaciones de cara al diagnóstico. Este punto es clave puesto que el objetivo final que se busca con el acceso a cualquiera de las dos bases de datos es la automatización del acceso a la información de cada patología analizada. Para ello, contar con datos que mantengan una estructura lo más uniforme posible es fundamental si se quiere implementar un proceso realmente autónomo.

- **eMedicine** ofrece acceso gratuito a toda su web mediante la creación de una cuenta, mientras que para UpToDate es necesario pagar una suscripción mensual. Aunque no determinante, es conveniente destacar también este punto.

Por tanto, la elección final ha sido la utilización de la web de **eMedicine**, puesto que nos ofrecía a coste cero prácticamente la misma información que la otra opción y mucho más estructurada. Entrando a la web, observamos que existe un campo de búsqueda y, a través de él, podemos acceder a cualquier patología de las que exista registro.

A modo de ejemplo, en la Figura 19 observamos la entrada que nos ofrece eMedicine al buscar sobre la Apendicitis. En el panel izquierdo, encontramos un índice de todos los apartados en los que se encuentra organizada la información y, a la derecha, aparece la información como tal. Esto es realmente importante puesto que, como ha sido comentado, este esquema se suele repetir en todas las entradas de la página web y, por tanto, podemos automatizar la descarga de datos en base a él.

Por último, tras consultar previamente con Ramón Puchades, médico internista del Hospital Universitario La Paz, de Madrid, (en adelante, el facultativo), se toma la decisión de apuntar al apartado de *Presentation* por ser éste el que mayor relevancia presenta de cara al proyecto a realizar. Dentro del mismo, se recopilarán los textos que se encuentran en los subapartados de ***History*** y ***Examination***.

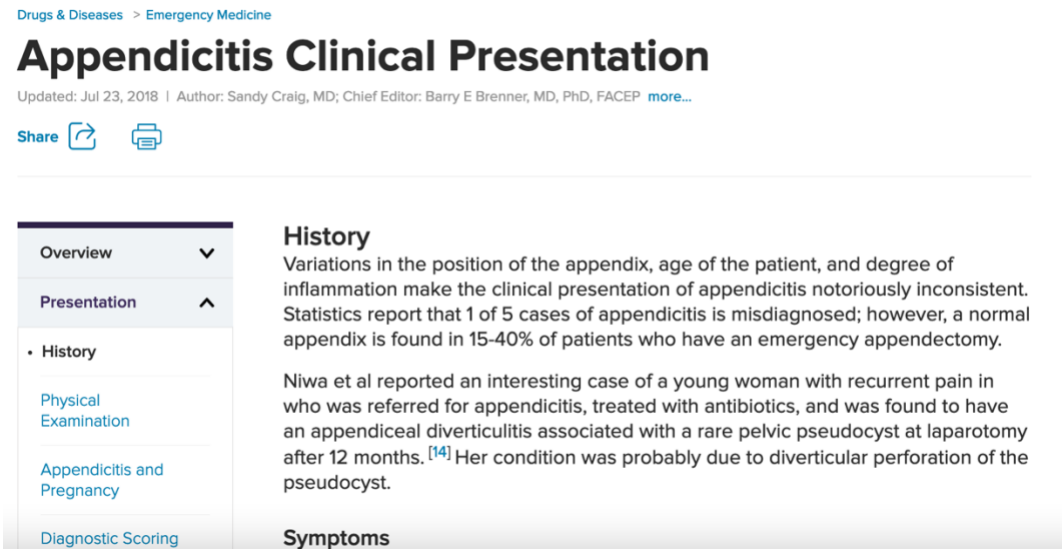

Figura 19 *Artículo en eMedicine*
Fuente: Portal oficial de eMedicine

3.1.3 Fuentes de información alternativas

Como se ha comentado en el punto anterior, utilizaremos un apartado en específico de la web de eMedicine. En la gran mayoría de casos, esto será suficiente, sin embargo, para algunas de las patologías a tratar o bien no existirá entrada en el portal la entrada para las mismas o bien no se encontrará dentro del índice dicho apartado.

Para estos casos especiales, se ha tomado la decisión de acceder a otras fuentes de información médica como artículos científicos, la American Academy of Family Physicians[8], Nature[9] o el National Center for Biotechnology Information[10].

Una vez identificados los artículos correspondientes a cada una de las patologías buscadas, de nuevo se tomará la opinión del facultativo para valorar que puntos de cada uno de los documentos resultan interesantes para el estudio y, además, se asemejan en mayor medida a los apartados obtenidos para el resto enfermedades mediante eMedicine, maximizando de esta forma la coherencia en los datos recopilados.

3.2 Equipamiento hardware

Para la realización de todas las actividades relacionadas con este proyecto, es decir, tanto las tareas de investigación computacional, como el procesado de datos y el entrenamiento de los modelos, entre otras, ha sido utilizado un *MacBook Pro de 13"* con un *procesador Intel i5 de doble núcleo a 2,9 GHz, 8 GB de memoria RAM* y *tarjeta gráfica integrada Intel Iris Graphics 550*.

Al inicio de la realización del trabajo, se planteó la posibilidad de trabajar sobre la plataforma online de Google llamada *Google Colaboratory*[11]. Esta herramienta online permite la ejecución de código Python utilizando el hardware de Google en la nube, lo cual suele acelerar los procesos respecto a la ejecución en local de los mismos. Sin embargo, el contrato de confidencialidad firmado para la utilización de la base de datos del 112 impedía su uso fuera de un entorno académico controlado y seguro, por lo que la idea fue desechada.

En cualquier caso, los cálculos realizados no realizan un consumo elevado de recursos, por lo que la utilización de un dispositivo personal ha sido más que satisfactoria, con la excepción de la traducción, problema que se comentará más adelante.

3.3 Equipamiento software

3.3.1 Lenguaje de programación y entorno de desarrollo

En el desarrollo del presente trabajo, para todo el código generado durante la creación del web «scraper», el procesado de datos y el entrenamiento de los distintos modelos ha sido utilizado

[8] Web oficial: https://www.aafp.org
[9] Web oficial: https://www.nature.com
[10] Web oficial: https://www.ncbi.nlm.nih.gov
[11] Web oficial: https://colab.research.google.com

como lenguaje de programación Python, en concreto, la versión 3.8.5. La motivación de esta elección viene dada por dos aspectos principales:

- **Curva de aprendizaje**: El lenguaje es realmente sencillo de aprender y en pocas horas se puede empezar a programar en él con relativa soltura. Además, la existencia de muchos recursos bibliográficos facilita el proceso [24].

- **Librerías**: En la actualidad, Python es el lenguaje dominante en el mundo del Machine Learning y uno de los motivos principales es el enorme ecosistema de librerías del que dispone. La instalación de estas es rápida y simple y la oferta enorme tanto en cantidad como en utilidad [25].

Una vez ha sido elegido el lenguaje de programación, es necesario decidir el entorno de desarrollo. En este caso, toda la parte de código ha sido escrita empleando ***Jupyter***[12] un IDE (Interactive Development Environment) de código abierto y basado en la web que permite ejecutar de forma rápida y sencilla líneas de código además de poder visualizar el resultado de forma instantánea. La forma de trabajo es mediante «*notebooks*» que permiten introducir tanto código como texto plano en celdas, permitiendo organizar el documento de forma más visual respecto a un documento de código plano.

A su vez, para intercambiar con el médico los patrones de enfermedades y ejemplos de pacientes virtuales, así como guardar las distintas estructuras de datos (o «*dataframes*») utilizadas a lo largo del trabajo ha sido empleado formato compatible con el popular programa *Excel*, puesto que permite visualizar las tablas de datos de forma sencilla a prácticamente todo el mundo debido al gran porcentaje de distribución que presenta.

3.3.2 Librerías empleadas

Como se ha comentado en el punto anterior, una de las principales ventajas de utilizar Python es la enorme cantidad de librerías de las que se dispone. Para este proyecto han sido empleadas algunas de ellas, de las cuales aparecen, a continuación, enumeradas y explicadas de forma breve las más importantes:

- **Pandas**: Esta librería es popularmente conocida por permitir el manejo de datos de forma sencilla. Con una llamada es capaz de leer ficheros en formato CSV y Excel, entre otros, y crear un objeto de tipo DataFrame con el que se puede manipular los datos fácilmente. A través de esta estructura de datos propia de la librería, tareas de análisis, filtrado, modificación y preparación sobre los datos se simplifican en gran medida. A su vez, permite almacenar los datos en formatos como los mencionados anteriormente. [*https://pandas.pydata.org*]

- **Requests**: Esta librería permite trabajar desde Python con peticiones HTTP, de forma que las mismas sean simples y amigables de realizar. Las peticiones como GET y POST se pueden realizar de forma rápida a través de funciones predefinidas

[12] Web oficial: https://jupyter.org

con unos pocos parámetros, por lo que es realmente útil para tratar realizar llamadas a APIs externas y webs. [*https://docs.python-requests.org/en/master/*]

- **BeautifulSoup**: Una vez realizada la llamada HTTP, en caso de recibir respuesta ésta será un documento HTML que deberá ser analizado. Esta librería ofrece la posibilidad de extraer la información necesaria de respuestas en formato HTML o XML. Por ello, es realmente útil en tareas de web scraping, siendo una de las más utilizadas para dicho objetivo. [*https://pypi.org/project/beautifulsoup4/*]

- **Regex**: En el tratamiento de palabras, destaca esta librería por permitir incrustar un lenguaje de programación especializado para definir patrones dentro de cadenas de caracteres. Gracias a ella, se pueden definir expresiones regulares para eliminar, sustituir o analizar cadenas de caracteres dentro de textos más largos. [*https://docs.python.org/3/library/re.html*]

- **NLTK**: El Natural Language Toolkit es la plataforma líder dentro de Python para el tratamiento del lenguaje natural. Incorpora recursos como WordNet para clasificar familias de palabras o un conjunto de *stopwords* que permite eliminar palabras que no son necesarias en el análisis de textos. Por ello, junto a otras funciones, destaca como herramienta para la clasificación de textos, la tokenización de palabras y en el proceso de Stemming y Lemmatization. [*https://www.nltk.org*]

- **Deep-Translator**: Existen numerosas librerías para Python que añaden la funcionalidad de traducir palabras y textos. Sin embargo, Deep-Translator destaca por permitir, con una única librería, traducir textos empleando diferentes recursos de online como Google Translator, Linguee o DeepL. Finalmente, para este proyecto ha sido empleada la traducción de Google sobre el resto por razones de velocidad, límite de caracteres y precio. [*https://pypi.org/project/deep-translator*]

- **Scikit-learn**: Como ya se ha comentado, Python destaca como lenguaje para el machine learning, de forma principal, por las librerías para dicho lenguaje. Una de las más conocidas y útiles es Scikit-learn que permite de forma simple y rápida generar una gran variedad de modelos tanto de clasificación como regresión y clusterización mediante distintos algoritmos. Además, aunque existen otras librerías más especializadas, ofrece también la utilización de pequeñas redes neuronales. [*https://scikit-learn.org/stable*]

Además de la utilidad de todas y cada una de las librerías mencionadas, la elección de las mismas en detrimento de otras parecidas o similares ha venido dado por su simplicidad y por ser, todas ellas, software open-source, lo cual supone un coste cero, una gran estabilidad y la opción de utilizar dicho software de forma totalmente libre.

4 Métodos

En este capítulo realizaremos la explicación de la metodología seguida en el proyecto, así como de los pasos realizados y la motivación de las decisiones tomadas. Para facilitar la comprensión de los procedimientos seguidos y estructurar de forma visual el conjunto de metodología seguida, en la Figura 20 podemos encontrar un esquema gráfico de los pasos seguidos.

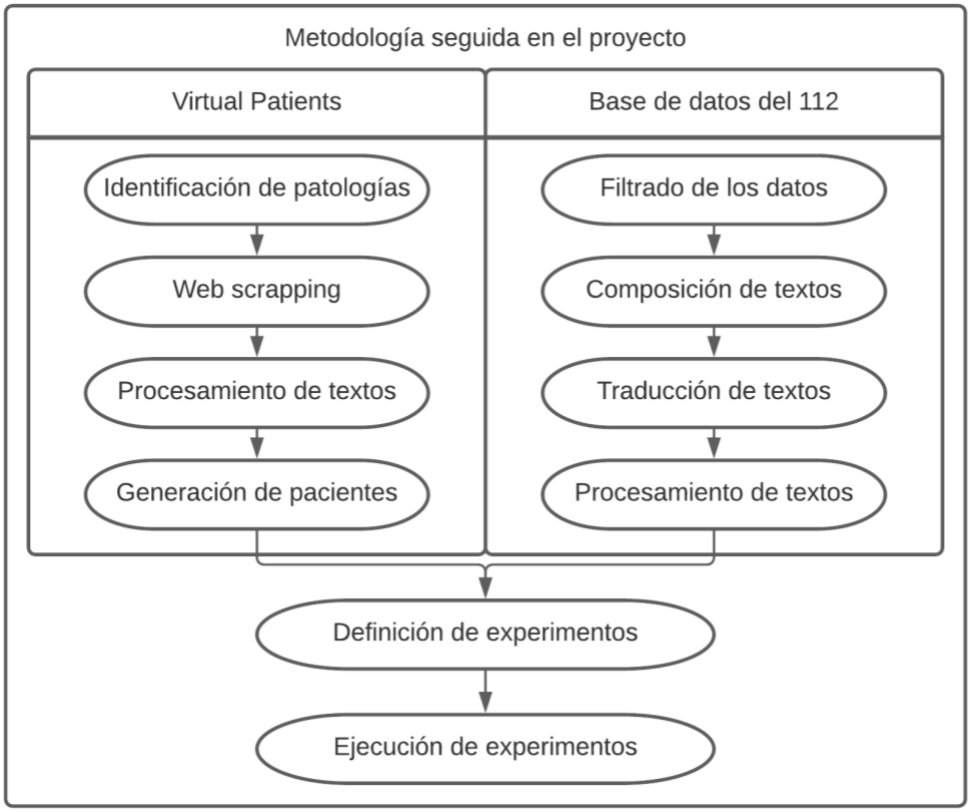

Figura 20 *Esquema de la metodología seguida*
Fuente: Elaboración propia

4.1 Identificación de patologías

Como ya ha sido mencionado anteriormente, la intención que se persigue con el presente proyecto es doble: por un lado, diseñar y crear un generador de pacientes virtuales y, por otro, el

diseño de un clasificador que permita evaluar la existencia de riesgo vital o no para un caso clínico dado durante misiones de exploración espacial. La enorme extensión en la variedad de patologías distintas que puede experimentar el ser humano hace que abordar un trabajo sin límites en cuanto a qué ramas o aspectos de la medicina se deben incluir en el estudio sea realmente complejo.

Por este motivo, para la realización de esta tarea, la NASA ha trabajado en la identificación de las posibles condiciones médicas más probables en la exploración espacial, lo que nos servirá para acotar el número de patologías a estudiar en el TFG de forma que el resultado siga siendo relevante a la vez de hacer factible el proceso. Por ello, teniendo en mente que el propósito final del trabajo es su aplicación para viajes espaciales y, por tanto, sabiendo que se encuentra orientado a astronautas, el punto de partida han sido las **cien patologías más comunes** que se pueden dar en el espacio exterior. Este conjunto de patologías viene dado por el plan de prevención de riesgos de la NASA publicados en [1], las cuales podemos encontrar en la Figura 18.

Una vez ha sido obtenido el listado de patologías a tratar, es necesario realizar dos tareas. En primer lugar, la asignación de los códigos CIE no solo es necesaria para tratar con la base de datos de las llamadas del 112, sino que facilita la búsqueda de información y estandariza el tratamiento de las condiciones médicas a nivel internacional. En segundo lugar, como ya ha sido mencionado, es ineludible obtener información de cada patología para compensar la carencia de casos reales en la base de datos. Para ello se utilizará el portal de eMedicine.

Ambos procesos pueden ser realizados por cualquier persona con conocimientos básicos de medicina y el uso de tiempo para las labores de investigación y comprobación. No obstante, para este proyecto la colaboración del facultativo ha refinado el proceso, pues ha sido el encargado de indicar los códigos CIE a utilizar y la revisión de las fuentes bibliográficas elegidas por el autor del trabajo. A su vez, han sido añadidos algunos diagnósticos más a las condiciones médicas que, a juicio del facultativo, podían resultar relevantes al estar estrechamente relacionadas con las descritas por la NASA y ser frecuentes o interesantes para añadir más información (v. g., a la «infección pulmonar» se le ha añadido la patología médica de «neumonía»).

Se puede encontrar en el Anexo I (Apartado 9.1) el archivo «*00_CIE_emedicine_info.xlsx*», en el que se muestra el resultado de este proceso. Respecto a este archivo, cabe destacar dos cosas:

- En primer lugar, la ausencia de valor en la celda de la columna de **CIE** para dos condiciones médicas se explica por la ausencia de un valor único para las mismas. Sería necesaria más información para poder decidir cuál de ellos aplicar.

- En segundo lugar, se puede observar que existen diversas celdas en la columna de **WEB** que se encuentran vacías. Como ya se ha explicado, ha sido necesario el uso de fuentes de información alternativas para estos casos, pues que la respuesta de eMedicine o bien no era satisfactoria o bien inexistente.

4.2 Web scraping

Una vez han sido recopiladas las fuentes de las que se extraerá la información (bien sea eMedicine o las alternativas ya mencionadas) es el momento de automatizar la extracción de la información. Para ello, ha sido creado un algoritmo de *Web Scraping* a medida de las necesidades del presente proyecto que podemos ver resumido en la Figura 21. Su código se encuentra en el Anexo 2 (Apartado 9.2) y referencia al archivo « *_01_scrape_emedicine.ipynb* ».

Para la realización del algoritmo han sido necesarias las librerías de *Requests* para realizar las peticiones GET y de *BeautifulSoup* para poder acceder al archivo HTML de cada una de las webs enlazas. De forma previa, se ha realizado un análisis de la estructura de la web en el que se ha observado que los apartados de interés para el proyecto («*History*» y «*Examination*») se encuentran ubicados dentro de un div (una etiqueta HTML que permite crear secciones) con un identificador del tipo '*content_bx'* y con una clase con valor '*refsection_content*'. Las funciones *get_online_info* y *get_page* se encargan de obtener esta información con un intervalo de tiempo de entre cinco y quince segundos para evitar que la web detecte actividad sospechosa en nuestro software.

Además, al recopilar texto del portal, éste mantiene los saltos de línea necesarios para la interfaz del dominio que, sin embargo, de cara a su posible lectura resultan innecesarios y molestos. Por ello, la función *trim_text* se encarga de eliminar todos aquellos conjuntos de dos o más saltos de línea, dejando únicamente uno.

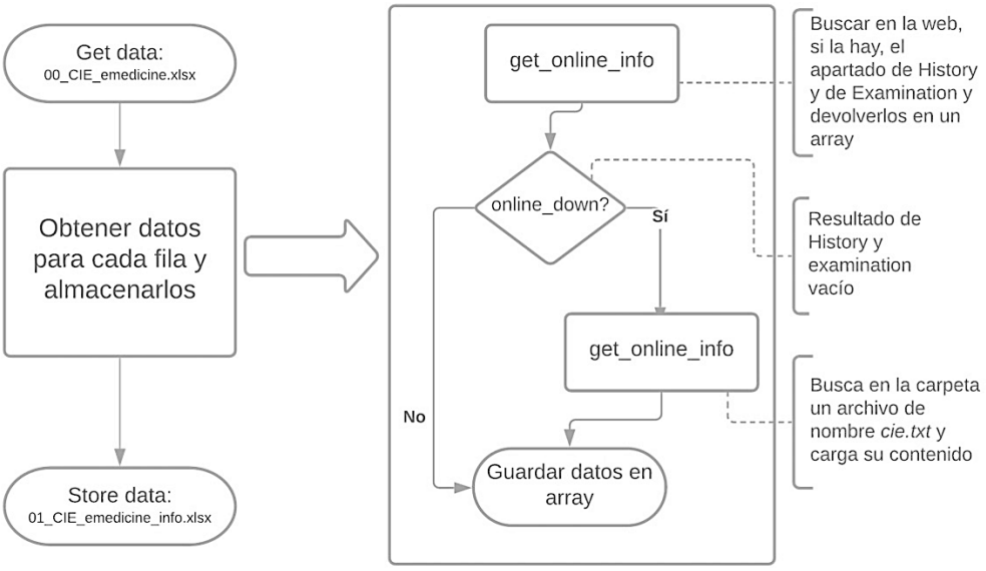

Figura 21 *Diagrama del algoritmo de web scraping*
Fuente: Elaboración propia

A su vez, la web puede no existir o incluso existir, pero no tener la estructura que se busca por cualquier motivo y, por tanto, las variables de («*History*» y «*Examination*» quedarían vacías, lo cual se comprueba con la función *online_down*. Por este motivo se ha creado una estructura de datos en la carpeta «*data/offline_data*» en la que se almacenarán archivos .txt con el CIE en cuestión como nombre del archivo a partir de las fuentes de datos alternativas.

A modo de resumen y como se observa en el diagrama de la figura anterior, el funcionamiento del algoritmo se basa en la iteración sobre el conjunto de filas (en este caso, el archivo del Anexo I) en el que se espera recibir un enlace a una web, recopilar la información buscada, eliminar los saltos de línea sobrantes y almacenar los datos. En caso contrario, se obtendrá la información previamente introducida en archivos individuales, como ya se ha explicado.

El resultado de toda esta fase es el archivo «*01_CIE_emedicine_info.xlsx*» en el que, a las columnas anteriores, se les añaden tres: «**HISTORY**» y «**EXAMINATION**» en caso de recopilar la información de la web y «**GLOBAL**» en caso de ser a partir de las fuentes adicionales.

A modo de ejemplo, describiremos el resultado obtenido para la condición médica de «Hernia abdominal». En este caso, accediendo a la web asociada que se muestra en la Figura 22, podemos ver que en el apartado de «History» se encuentran una serie de párrafos intercalados por imágenes y publicidad. Sin embargo, una vez llevado a cabo el proceso descrito anteriormente, el resultado obtenido será un único texto con todos los párrafos de la página unificados como el siguiente:

«In an emergency setting, a patient with a hernia may present because of a complication associated with the hernia, or the hernia may be detected on routine physical examination. (…)
Because this hernia is usually asymptomatic, patients typically present with a bulge at the site of a previous incision; the lesion may become larger upon standing or with increasing intra-abdominal pressure»

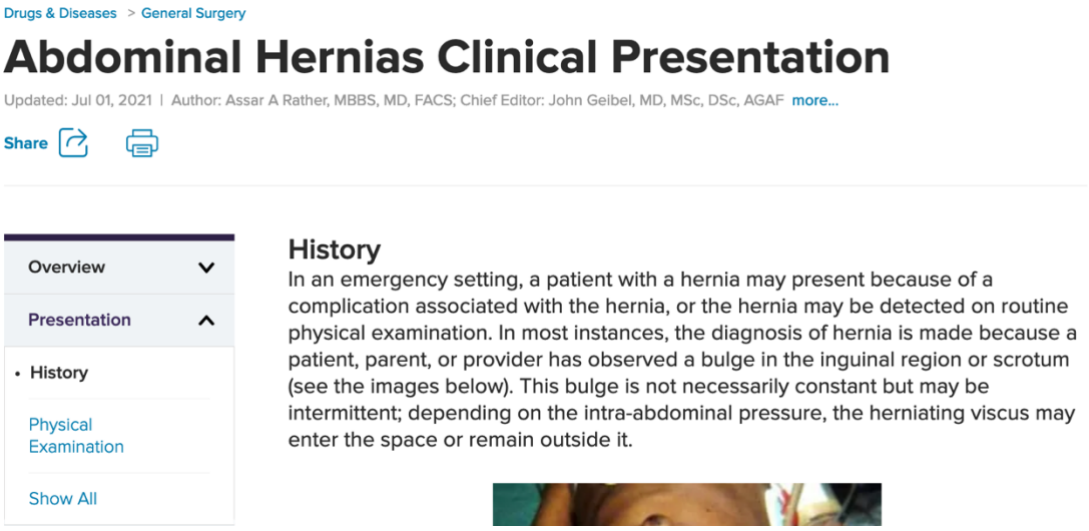

Figura 22 *Entrada de eMedicine para la hernia abdominal*
Fuente: Portal eMedicine - *https://emedicine.medscape.com/article/189563*

4.3 Procesamiento de textos

Una vez la información de las condiciones médicas seleccionadas ha sido almacenada, es momento de realizar el análisis y procesado correspondiente. El código de todo este procedimiento se encuentra dentro del archivo «*_02_word_analysis.ipynb*» junto al del procesado de la base de datos del 112, el cual será explicado en el siguiente apartado. Por ello, el contenido del archivo se ha dividido en dos Anexos, pudiendo encontrar lo relativo a este punto en el Anexo III (Apartado 9.3).

Tal y como se ha visto anteriormente, el pre-procesado de textos para su uso en Machine Learning consta de cinco fases bien diferenciadas. Aunque en este punto no se va a aplicar ninguna técnica de Aprendizaje Automático, vamos a realizar las cuatro primeras de estas cinco etapas para procesar el texto puesto que, posteriormente, será utilizado en la generación del texto libre de los pacientes virtuales[13].

[13] Para ilustrar el flujo llevado a cabo en este proceso usaremos como ejemplo la frase «*[27] : Swelling or fullness at the hernia site.*» extraída del texto de la patología de Hernia abdominal.

Figura 23 *Fases realizadas del preprocesado de datos*
Fuente: Elaboración propia

En primer lugar, cada uno de los textos descargados para cada condición médica debe ser tokenizado. Para ello, la función *tokenize_this* se encarga de pasar todas las letras a minúsculas y eliminar tanto los números como los caracteres que no son relevantes para el análisis de texto. Una vez hecho esto, la misma función devuelve, en lugar de un string, una lista de strings siendo cada elemento una palabra[14]. Cabe destacar que, al representar el texto lenguaje natural, no elimina los elementos repetidos, pues se pretende mantener la distribución y frecuencia de palabras.

Tras haber sido tokenizado el texto, la función *tokenize_and_filter* se encarga de eliminar las palabras que carecen de significado como tal y que son frecuentemente usadas en el lenguaje (tales como preposiciones o determinantes)[15]. Para ello, ha sido utilizado el conjunto ofrecido por la librería de NLTK, aunque como vemos en la función *get_tokenized_words* han sido añadidas al mismo una serie de palabras extra obtenidas de la web [https://www.ranks.nl/stopwords] que tampoco presentan utilidad y que hacen pasar el conjunto de 179 a 858.

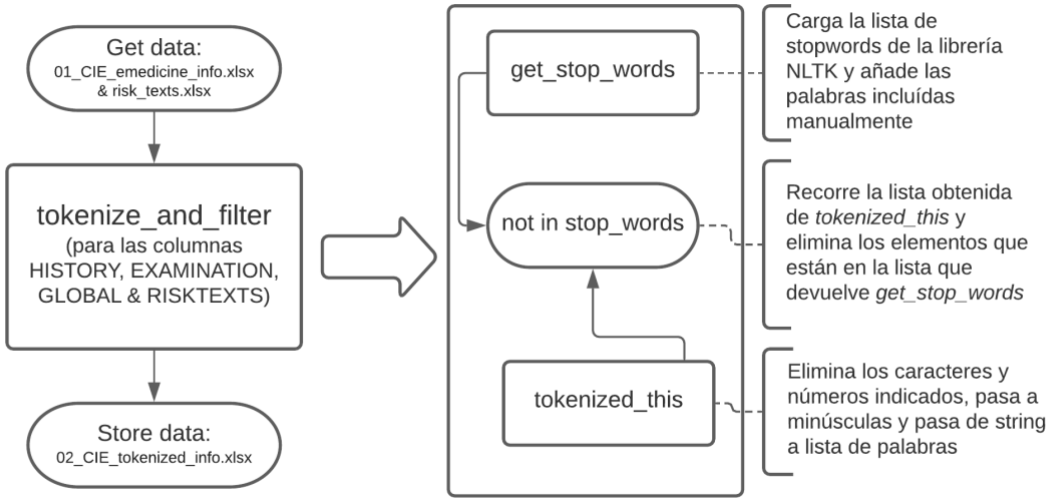

Figura 24 *Procesado de palabras del proyecto*
Fuente: Elaboración propia

El proceso descrito en los párrafos anteriores se realiza para las columnas de «**HISTORY**», «**EXAMINATION**» y «**GLOBAL**» para, finalmente, unificar las tres listas en una única que se guarda en la columna «**TOKENIZED**».

[14] En el caso de nuestro ejemplo, el resultado tras este proceso sería la lista *['swelling', 'or', 'fullness', 'at', 'the', 'hernia', 'site']*.

[15] Por último, el resultado final sería la lista *['swelling', 'fullness', 'hernia', 'site']*.

Tal y como se verá en el Apartado 4.5, la decisión de considerar un caso clínico como de riesgo vital o no vendrá dado por la existencia, dentro del texto libre, de alguna palabra marcada como indicadora de posibilidad de éxitus. Es decir, es necesario determinar, del conjunto de palabras obtenido para cada condición médica, aquellas que denotan riesgo. Para ello, ha sido realizada una revisión manual, asistida por el facultativo, por el que se han seleccionado aquellas palabras que presentan riesgo vital para cada diagnóstico. Este subconjunto de palabras se puede encontrar en el archivo «*risk_texts.xlsx*» como texto plano, por lo que antes de guardarse en el DataFrame se pasarán sus celdas por la función *tokenize_and_filter*. Para ilustrar el concepto de estas palabras hemos rescatado las escogidas para la lesión abdominal: *['emergency', 'stab', 'wounds', 'injury', 'accident', 'trauma', 'chest', 'vital', 'resuscitation', 'unstable']* que, como se puede apreciar, hacen referencia tanto a la sintomatología como al conjunto de circunstancias que pueden indicar riesgo vital en el paciente.

Por último, el resultado de todo este proceso se puede encontrar almacenado en el archivo «*02_CIE_tokenized_info.xlsx*», cuyo esquema se puede visualizar en la Tabla 2. En él, han sido almacenadas un total de 40.605 tokens, de las cuales se pueden extraer un total de 7.193 términos únicos.

NÚMERO	Número asociado a la condición médica en [1]
MEDICAL CONDITION	Nombre de cada una de las condiciones médicas estudiadas
CONDICIÓN MÉDICA	Traducción de los valores anteriores a español
CIE	CIE asociado a la condición médica
TOKENIZED	Lista con el conjunto de palabras obtenidas tras el procesado de los textos de HISTORY, EXAMINATION y GLOBAL
RISKWORDS	Lista con las palabras obtenidas tras el procesado del texto con las palabras que presentan riesgo

Tabla 2 *Estructura del archivo 02_CIE_tokenized_info.xlsx*
Fuente: Elaboración propia

4.4 Procesado y traducción de los casos reales del 112

Tras explicar el procesamiento llevado a cabo sobre los datos extraídos de las patologías a analizar, en este punto se analizará la **transformación a efectuar** sobre la base de datos original de las llamadas al 112, la cual ya ha sido expuesta en el **Apartado 3.1.1**. A su vez, las líneas de código del archivo «*_02_word_analysis.ipynb*» referentes a este proceso han sido incluidas en el Anexo IV (Apartado 9.4), pudiéndose observar en la sección de «*Get Data*» cómo son únicamente seleccionadas aquellas columnas o variables que se requieren en el presente trabajo, así como la unificación de la mayoría de ellas en la columna «**TEXT_ES**».

Una vez obtenidas las variables correspondientes al CIE, edad, sexo y texto libre, es preciso realizar el mismo proceso que ha sido expuesto en el Apartado anterior. Estos textos, empero, se encuentran escritos en español, por lo que, tras pasarlos a minúsculas, eliminar los caracteres

innecesarios, tokenizarlos y en este caso devolverlos como un string único, el siguiente paso será traducirlos al inglés.

Con dicho propósito y como ya ha sido adelantado previamente, se ha utilizado la traducción de Google mediante la librería Deep-Translator, puesto que es completamente gratuito, traduce de forma fiable y el límite de caracteres se encuentra en 5.000 por llamada. Sin embargo, este paso presenta un problema: las llamadas al traductor son relativamente lentas, lo cual no supone un problema para traducir un par de textos, pero al hablar de 136.160 casos distintos, la magnitud de tiempo requerido superaba las 48 horas, lo cual era difícil de manejar.

Por este motivo y para reducir el tiempo de trabajo necesario, se toma la decisión de agrupar los textos en grupos de veinte (evitando, así, superar el límite de los 5.000 caracteres en todas las llamadas) estableciendo como separador el salto de línea. De esta forma, tal y como se puede ver en la Figura 25, mediante la función *list_es_to_en* los textos son agrupados, traducidos y divididos de nuevo para asignarlos a cada paciente original, consiguiendo, así, reducir el tiempo de ejecución a menos de tres horas.

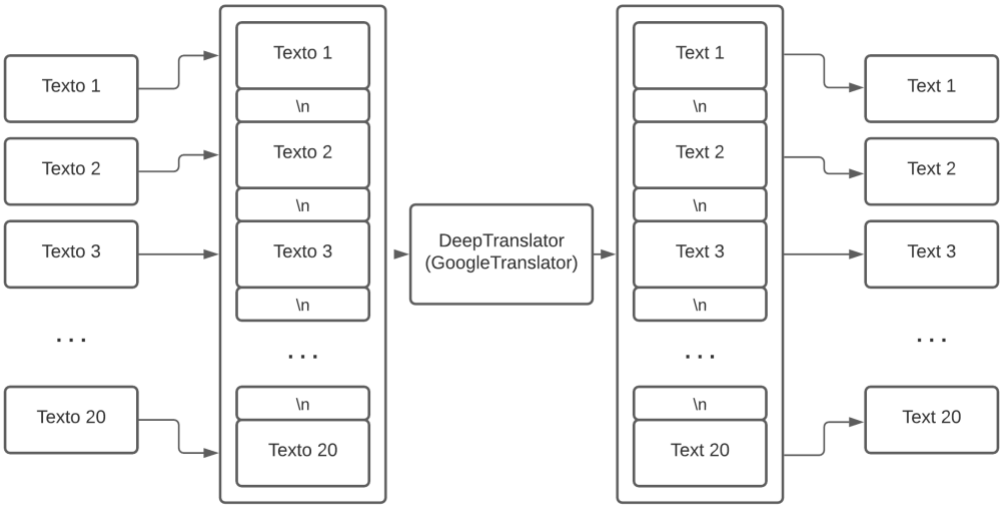

Figura 25 *Proceso de (des)agrupación para la traducción*
Fuente: Elaboración propia

Concluido este paso, los textos obtenidos en inglés sí pueden someterse al filtrado de las *stopwords*, para lo que se deberán volver a tokenizar previamente. Por último, se transformará de nuevo la lista a un string al haber terminado su pre-procesado, obteniendo, ahora sí, el conjunto de textos libres listos para la generación de modelos.

Es importante mencionar que, a pesar del proceso de agrupación realizado, tanto el tokenizado y filtrado como la traducción requiere de un tiempo considerable (en especial el proceso de comunicarse con Google Translator). Por ello, ha sido añadido a la función de *tokenized_words_to_str* un contador a imprimir cada 10.000 iteraciones y en la función de *list_es_to_en* se imprime cada un valor por cada una de las 6.808 llamadas a realizar.

Por otra parte, tal y como se mencionó al describir la base de datos, la codificación utilizada por el 112 usa, para el riesgo, valores de 1 y 2 y para la edad los días naturales. Por tanto, además del procesado de los datos se aplicará una función lambda para almacenar el valor previo del riesgo menos uno en formato *integer* y la función *days_to_year* para traducir los días en años, aunque debido a la existencia de edades incoherentes dentro el mismo paciente, en algunos casos

(en los que exista más de una edad) se devolverá el valor -1. Finalmente, los casos clínicos, acompañados de los textos en español e inglés, se almacenarán en el archivo «*02_112_patients_db.xlsx*».

Asimismo, una vez tenemos la base de datos del 112 incorporada al proyecto es posible recoger estadísticas de la misma y ver el número de pacientes que existen para cada CIE. Esto será relevante puesto que las patologías de las que verdaderamente interesa obtener pacientes virtuales son de aquellas cuya frecuencia en la base de datos de casos reales sea baja.

La verdadera necesidad de generar pacientes de un subconjunto de CIEs unido a la dificultad de revisar manualmente los más de 40.000 términos obtenidos de las cien patologías estudiadas, fuerza a tomar la decisión de analizar las palabras que presentan riesgo vital de las veinte patologías de las que menos casos existen en la comentada base de datos, pudiendo ser encontradas éstas en el archivo «*risk_texts.xlsx*».

4.5 Generación de Pacientes Virtuales

Tras obtener la información necesaria de las distintas patologías a estudiar y realizar el pertinente pre-procesado de los textos obtenidos, es el momento de definir cómo van a ser usados todos estos datos en la generación de los pacientes virtuales.

Para ello, se deben tomar una serie de decisiones en torno a las cuatro variables básicas que van a definir al paciente virtual [20]. El código asociado a todo este proceso se encuentra en el archivo «*_03_virtual_patient_generator.ipynb* », aunque a lo largo de los siguientes puntos se mostrarán las funciones esenciales, destacando la Figura 26 como la función encargada de generar el paciente virtual como tal.

```
def create_virtual_patient(cie):
    risk = random.choices([1, 0],
          weights = [probability_risk, 1 - probability_risk], k = 1)[0]
    return {
        "CIE": cie,
        "AGE": random.randint(min_age, max_age),
        "SEX": get_sex(cie),
        "RISK": risk,
        "TEXT": get_new_text(cie, risk),
    }
```

Figura 26 *Función para generar pacientes virtuales*
Fuente: Archivo «_03_virtual_patient_generator.ipynb»

Cabe mencionar que el generador de pacientes virtuales emplea una serie de parámetros de configuración obtenidos a partir de un archivo en formato *yml*, el cual se puede ver en la Figura 27. De esta forma, algunos de los valores de los que hablaremos a continuación se pueden modificar sin alterar el código, lo cual aumenta la reusabilidad del código de cara a futuros usos del software generado.

```yaml
patient:
    min_age: 20
    max_age: 45
    probability_man: 0.5
    probability_risk: 0.5
    min_length: 15
    max_length: 30
    just_men:
      - "601"
    just_women:
      - "112,1"
other:
    total_cases: 3000

```

Figura 27 *Archivo de configuración para la generación de Pacientes Virtuales*
Fuente: Archivo «config.yml»

4.5.1 Elección del riesgo

En primer lugar, es necesario saber si el paciente a generar presentará riesgo vital o no. Para ello, se emplea la variable de entorno «*probability_risk*», la cual define la probabilidad de que el paciente en desarrollo sea de riesgo. Gracias a que el número de pacientes a generar es elevado (en el caso de estudio, se ha asignado una cifra de 3.000 a dicho valor) podemos tomar esta probabilidad como el porcentaje de casos finales que presentarán riesgo en la muestra generada.

Cabe destacar que la codificación del riesgo será de valor 0 para su ausencia y de 1 para los casos en los que sí que exista.

4.5.2 Elección de la edad

La edad del paciente virtual es una variable que se definirá de forma bastante arbitraria. Hay que tener en consideración que el objetivo final de este proyecto es su aplicación a la exploración espacial, por lo que suponemos que las edades de uso están acotadas a los rasgos actuales de selección de astronautas [26]. Por ello, se selecciona en cada caso una edad entre 20 y 45 años con distribución uniforme. De nuevo, estos valores se obtienen como variables de entorno por lo que se pueden modificar fácilmente.

4.5.3 Elección del sexo

Respecto al sexo, la generación del mismo es bastante parecida a la de la edad: No obstante, en este caso se añade la dificultad de que no todas las patologías pueden aparecen en ambos (v. g., la Prostatitis o la Candidiasis vaginal). Por este motivo, la función generadora tiene restricciones para asignar de forma automática el sexo en caso de corresponderse el CIE a alguna de las patologías que se encuentran en las listas proporcionadas a través del mencionado archivo de configuración como de un determinado sexo.

```
def get_sex(cie):
    if cie in just_women:
        return 'M'
    else if cie in just_men:
        return 'H'
    return random.choices(['M', 'H'], weights=[probability_man, 1 -
probability_man], k=1)[0]
```

Figura 28 *Función para general del sexo del paciente virtual*
Fuente: Archivo «_03_virtual_patient_generator.ipynb»

4.5.4 Generación del texto libre

La generación del texto libre es la variable por asignar que presenta mayor dificultad, aunque es bastante consistente con lo explicado hasta ahora.

En primer lugar, al crear el DataFrame con los listados de palabras obtenidos en los pasos anteriores, lo primero será discernir entre las palabras que presentan riesgo y las que no. Para ello se utiliza la función *delete_risk_tokenized* que elimina de la columna «**TOKENIZED**» las palabras que aparecen en la columna «**RISKWORDS**», dejando el listado resultante en la columna «**NONRISKWORDS**».

Tras este proceso las palabras se encuentran claramente diferenciadas, por lo para generar cada paciente virtual es necesario obtener tanto el conjunto de palabras que presenta riesgo como las que no. Acto seguido, en base a dos variables de entorno que representan la longitud mínima y máxima del texto a generar se asigna dicho valor mediante una distribución uniforme.

Con la longitud del texto definida, en caso de tener que generar un paciente que no presenta riesgo el número de palabras seleccionadas del conjunto de riesgo será cero. En caso contrario, entre una y la mitad de la longitud del texto. En cualquier caso, hasta llegar a la longitud asignada el texto se rellenará con palabras del grupo de NO riesgo.

Es importante señalar que en ningún momento se establecen máximos de longitud en función del número de palabras disponibles puesto que la función *choices* permite la repetición de las mismas tal y como ocurre en el lenguaje natural, por lo que esto no supone un problema.

Por último, se ordena la lista de palabras de forma aleatoria y se devuelven en formato string.

```
def get_new_text(cie, risk):
    ordinary_words = cie_data.loc[cie_data['CIE'] == cie]['NONRISKWORDS'].iloc[0]
    risk_words = cie_data.loc[cie_data['CIE'] == cie]['RISKWORDS'].iloc[0]

    text_length = random.randint(min_length, max_length)
    number_risk_words = random.randint(risk, int(text_length/2) * risk)

    text = random.choices(risk_words, k = number_risk_words)
    text += random.choices(ordinary_words, k = text_length - number_risk_words)

    random.shuffle(text)
    return " ".join(map(str, text))
```

Figura 29 *Función para generar texto libre del paciente virtual*
Fuente: Archivo «_03_virtual_patient_generator.ipynb»

Terminado todo este proceso, el paciente virtual ha sido generado. Solo resta automatizar la generación en masa de los mismos. Para ello, la función *get_cies_with_risk_words* devuelve una lista con los CIEs de las condiciones médicas que tienen almacenadas palabras con riesgo, la función *create_df_patients* genera el conjunto de pacientes virtuales para cada uno de los CIEs pasados como argumento en una lista y, por último, la función *create_patients* genera el número de pacientes asignados mediante la variable de entorno para un CIE en concreto.

A título de ejemplo, mostramos a continuación dos casos clínicos obtenidos mediante el generador de pacientes virtuales para la patología de «Distensión de cadera». Por un lado, para el caso de NO riesgo, un paciente tipo sería el siguiente: «*hip neck study hours limited deep rest acting pulses patency persist leg area hip checking*». Por otro lado, en el caso de riesgo vital: «*rotation bone care fracture passive reveals heel hemorrhagic groin neck stand pain hip area rest hip*». Vemos como, en el segundo texto, la aparición de términos como «*fracture*», «*hemorrhagic*» o «*pain*» son términos identificados previamente como de riesgo, por lo que se clasifican como es esperado en base a la lógica definida.

De este modo, se almacena en el archivo «*03_virtual_patients_db.xlsx*» una base de datos de 60.000 pacientes virtuales de 20 patologías distintas y, por otro, 3.000 casos clínicos de la condición médica «Quemadura Solar» (CIE 692,76) en el archivo «*03_virtual_patients_cie_692,76.xlsx*».

4.6 Experimentos computacionales

Llegados a este punto, contamos con dos bases de datos (pacientes virtuales y pacientes reales) obtenidas mediante fuentes de información distintas, pero con una misma estructura y contenido adaptado a las condiciones médicas de exploración espacial. Por un lado, disponemos de los pacientes virtuales generados a partir de la información obtenida de las condiciones médicas más probables en los viajes espaciales y, por otro, un conjunto de 136.160 casos reales de las llamadas realizadas al 112 de la Comunidad Valenciana. Ambos conjuntos de datos se corresponden a los archivos «*03_virtual_patients_db.xlsx*» y «*02_112_patients_db.xlsx*» respectivamente.

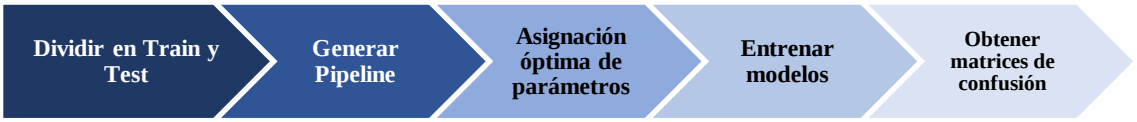

Figura 30 *Fases seguidas en los experimentos*
Fuente: Elaboración propia

Una vez delimitados los datos a utilizar, se procede a definir, como vemos en la Figura 30, la metodología común a todos los experimentos realizados. Para ello, en el archivo «*_04_ml_classification.ipynb*», el cual se puede visualizar en el Anexo V (Apartado 9.5), se incluye el esquema seguido en los mismos sobre el que se han ido realizando las modificaciones pertinentes para ajustarse a los experimentos. No obstante, se resaltarán los elementos más importantes en este mismo Apartado.

En primer lugar, será necesario discernir dentro de los datos a utilizar entre el subconjunto de entrenamiento y de prueba. Para ello, se ha tomado la decisión de utilizar el 75% de los datos para entrenamiento, dejando el 25% restante para evaluar el rendimiento de cada uno de los

modelos. Como se puede observar en la Figura 31 gracias a la función de Scikit-learn la colección de datos a emplear queda dividida en cuatro:

- **Xtrain, Ytrain**: En estas variables quedarán almacenados cada uno de los elementos a utilizar durante el entrenamiento. Por un lado, en la variable Y quedarán recogidos cada uno de los valores de las etiquetas de clase que definen la categoría de cada elemento. Por otro lado, en la variable X, se almacenará el resto de las variables de cada caso.

- **Xtest, Ytest**: Estos dos subconjuntos son análogos a los anteriores, pero en este caso se trata de los elementos a utilizar en la evaluación de los modelos.

```
X_train, X_test, y_train, y_test = train_test_split(
    virtual_patients.L_TEXT,
    virtual_patients.RISK,
    test_size = 0.25
)
```

Figura 31 *Función para dividir los pacientes virtuales en train y test*
Fuente: Archivo «_04_ml_classification.ipynb»

Una vez los conjuntos de datos con los que se va a trabajar han sido definidos, el siguiente paso será diseñar lo que en Scikit-learn se conoce como «***Pipeline»***. Este concepto hace referencia a un conjunto de operaciones que se realiza de forma encadenada, siendo los datos de entrada de cada función la salida de la anterior. El hecho de utilizar varios modelos y tener que realizar acciones de forma previa se simplifica utilizando dicha herramienta. Como se observa en la Figura 32 el *Pipeline* definido consta de tres acciones:

- **TFIDF**: La primera función a utilizar será la encargada de transformar los textos de entrada en la matriz TF-IDF directamente. Como se ha explicado en el Apartado 2.2.3, el ordenador no utilizará palabras como tal para realizar el entrenamiento, sino que transformará cada una de ellas en un peso asociado en función de su frecuencia y popularidad en el conjunto de textos, acorde a la fórmula de la Figura 14. A su vez, se puede observar que la función admite una serie de parámetros, cuyo uso será comentado en los ulteriores experimentos.

- **Select**: La segunda función permite seleccionar las *k* palabras con mejor resultado del conjunto de pesos que se devuelve en la operación anterior. Para nuestro caso, se ha tomado la decisión de utilizar el test *chi2* en todos los experimentos por su eficiencia y sencillez, aunque cabe destacar que el valor de *k* se determinará, de nuevo, en los distintos experimentos. De esta forma, a través del citado test se descartarán aquellos elementos que muestren una mayor probabilidad a ser independientes, pues son estos los casos que aportan menos información de cara a realizar la clasificación.

- **Model**: La tercera y última función a emplear será la que se corresponde con el modelo que se va a utilizar para entrenar los datos. Finalmente, debido a las características de los datos de entrada (conjuntos de palabras) y la voluntad de realizar una comparación entre diferentes algoritmos, se ha tomado la decisión de emplear los siguientes modelos: *MultinomialNB* (o Naive Bayes), *LinearSVC* (SVM con *kernel* lineal, como ya se había adelantado), *Random Forest* (con cien árboles y

boostrap activado) y *MLClassifier* (con cien unidades en una capa oculta, un optimizador para el cálculo del peso llamado «lbfgs» y un máximo de 200 iteraciones por defecto).

```
pipeline = Pipeline([('tfidf', TfidfVectorizer(...)),
                     ('selec',  SelectKBest(chi2, k='all')),
                     ('model', MultinomialNB())])
```

Figura 32 *Esquema para crear un pipeline en Scikit-Learn*
Fuente: Archivo «_04_ml_classification.ipynb»

Concluida la definición de los pipelines se ha procedido a utilizar una función propia de Scikit-learn que permite realizar modelos asignando diversos valores a un determinado parámetro, de forma que se escoja el que mejor resultado ofrezca. En nuestro caso, el atributo «*ngram_range*» se utiliza para entrenar el modelo empleando subconjuntos de uno o dos términos. Es decir, durante el entrenamiento se utilizarán, por un lado, las palabras de forma individual y, por otro, agrupadas en bigramas, siendo el modelo resultante el que mejor rendimiento presente de los dos. Gracias a ello, conceptos como *no bueno* pueden ser valorados. A su vez, con el parámetro «*alpha*» se asigna al término regularización el valor que maximiza el rendimiento.

Como ya ha sido mencionado previamente, el último de los cinco pasos del procesamiento del lenguaje natural no ha sido llevado a cabo. Este proceso de *lemmatization* se realizará utilizando la librería de NLTK y, en concreto, la función *WordNetLemmatizer*. Para ello se analizará cada una de las palabras para obtener su categoría gramatical y, mediante su clase y la palabra en sí, la función se encargará de devolvernos la raíz de la misma. No obstante, este proceso no se va a llevar a cabo para todos los experimentos, por lo que en la explicación de cada uno se indicará su uso.

De cara a preparar los datos a usar en los experimentos, a la hora de definir los conjuntos de *train* y *test* es imprescindible que el muestreo se realice de forma aleatoria. Sin embargo, esto hace que cada vez que se lleven a cabo los experimentos se obtengan subconjuntos distintos y, por tanto, resultados distintos. Para evitar este hecho y conseguir experimentos reproducibles, a la hora de definir las variables a utilizar se ha añadido el parámetro «*random_state*» con el valor 76. De esta forma, nos aseguramos de que los subconjuntos obtenidos sean siempre los mismos y, por tanto, los resultados también.

A su vez, para analizar y comparar los resultados se utilizará la llamada **matriz de confusión** y una serie de valores obtenidos a partir de la misma. Como podemos ver en la **Error! Reference source not found.**, ésta nos facilita los elementos utilizados como *test* clasificados en los que eran de no riesgo y cuya predicción ha sido no riesgo (TN del inglés True Negative), los casos que eran de riesgo y cuya predicción ha sido de riesgo (TP del inglés True Positive) y aquellos casos en los que la predicción falla (FP del ingles False Positive o predicción de riesgo cuando era de no riesgo y FN del inglés False Negative o predicción de no riesgo cuando era de riesgo).

	PREDICE NO RIESGO	*PREDICE RIESGO*
NO RIESGO	TN	FP
RIESGO	FN	TP

Tabla 3 *Matriz de confusión*
Fuente: Elaboración propia

Explicada la matriz de confusión, a partir de ella se obtendrán una serie de valores que nos permitirán estudiar la bondad de cada uno de los modelos:

- **RECALL-SENSIBILITY** (TP / [TP + FN]): Esta ratio se refiere a la capacidad del modelo de identificar los casos que presentan riesgo vital, es decir, devuelve el porcentaje de los casos en los que se predice riesgo de todos los que verdaderamente lo presentan.

- **SPECIFICTY** (TN / [TN + FP]): Este valor indica lo mismo que el anterior, pero para los casos de no riesgo. Es decir, devuelve el porcentaje de los que identifica como de no riesgo de entre todos los que verdaderamente no lo presentan.

- **PRECISION**: (TP / [TP + FP]): En este caso se obtiene el porcentaje de casos que verdaderamente presentan riesgo de todos en los que predice riesgo.

- **ACCURACY** ([TP + TN] / [TP + FP + FN + TN]): Esta métrica es quizás el más conocido y devuelve el porcentaje de aciertos del modelo, aunque en los casos en los que las clases no están balanceadas, es decir, cuando el número de casos de una clase es considerablemente superior al de otra, puede llegar a ser engañoso.

- **F1** (2 * [*Recall * Precision*] / [*Recall + Precision*]): Debido a los problemas que puede presentar utilizar conjuntos de datos no balanceados, se suele utilizar esta métrica que calcula la media armónica entre el Recall y el Precision , lo cual evita dichos sesgos.

- **SUPPORT**: Por último, cabe destacar, aunque no sea una ratio, la métrica de *support*, el cual nos indica el número de elementos que existen para cada clase. En nuestro caso, todas las bases de datos se encuentran muy balanceados, siendo el caso de los pacientes virtuales y la base de datos del 112 un 50–50% y un 55–45% para no riesgo–riesgo respectivamente.

Por último, es importante destacar que, según el contexto del problema, alguno de los parámetros anteriores puede ser más importante que el resto. Por ejemplo, en nuestro caso en el que se trata el tema de la salud, se priorizará maximizar el *recall* antes que el *precision*, puesto que será mejor clasificar como riesgo casos que no lo presentan que no identificar riesgos en los que pudiera haberlo. Para estos casos en los que ambas métricas no adquieren la misma importancia existen ratios distintas al F1 para dotar de más peso a uno de los dos, aunque, en este caso, al ofrecer todos los parámetros no se ha considerado necesario.

Analizado el proceso llevado a cabo para el entrenamiento de los modelos, el siguiente punto será definir las condiciones de cada experimento para, posteriormente poder observar y comparar los resultados obtenidos.

- **Experimento 1**: El primer experimento se llevará acabo con el conjunto de 3.000 pacientes asociados a la patología de quemadura solar (CIE 692.76) para comprobar el desempeño de los distintos modelos a utilizar con una sola condición clínica. Será llevado a cabo con el conjunto total de palabras, sin parámetros en la generación de la matriz TF-IDF, habiendo sido eliminadas las *stopwords* y sin realizar el proceso de *lemmatization*.

- **Experimento 2**: En el segundo experimento utilizaremos la base de datos con los 60.000 pacientes virtuales generados y las 136.160 llamadas del 112, aunque de forma separada. En este caso, la generación de la matriz TF-IDF será también sin parámetros, aunque sólo se utilizarán los k=1.000 términos más relevantes. Por último, seguiremos sin utilizar la *lemmatization* y se comprobarán los efectos de eliminar o no las *stopwords* de la base de datos del 112.

- **Experimento 3**: En este experimento, se mantendrán las condiciones del anterior con la excepción de la generación de la matriz TF-IDF y el número de términos a utilizar. Para el primer objetivo, se asignará a la variable «*min_df*» el valor de 5 (lo cual indica que sólo se utilizarán aquellos términos que aparecen en, al menos, cinco pacientes) y a la variable «*max_df*» 0,7 (lo que indica que se utilizarán los términos que aparezcan, como máximo, en el 70% de los casos). De esta forma, se eliminarán aquellos términos tan infrecuentes que no aportan información, así como los que aparecen en muchos pacientes,. Por último, asignando k=’all’ se utilizarán todos los elementos que cumplan las condiciones previas, independientemente del número que sea.

- **Experimento 4**: Este cuarto experimento conservará las condiciones del anterior, aunque, ahora sí, será llevado a cabo el proceso de *lemmatization* para los textos de ambas bases de datos.

- **Experimento 5**: Por último, en el quinto experimento, se llevará cabo una unificación entre ambas bases de datos, lo que resultará en un conjunto de 196.160 casos clínicos. A su vez, se llevará a cabo la creación de la matriz TF-IDF con los parámetros explicados previamente, utilizando todos los términos resultantes, eliminando las *stopwords* y aplicando el proceso de *lemmatization*.

A modo de resumen, en la Tabla 4 podemos encontrar recopilada la casuística de cada uno de los experimentos llevados a cabo.

	Base de datos unificada	K=	Uso de parámetros en TF-IDF	Stopwords eliminadas	Lemmatization
Experimento 1	-	ALL	NO	SÍ	NO
Experimento 2	NO	1.000	NO	NO/SÍ	NO
Experimento 3	NO	1.000	SÍ	SÍ	NO
Experimento 4	NO	1.000	SÍ	SÍ	NO
Experimento 5	NO	1.000	SÍ	SÍ	SÍ
Experimento 6	SÍ	1.000	SÍ	SÍ	SÍ

Tabla 4 *Resumen de los experimentos llevados a cabo*
Fuente: Elaboración propia

4.7 Página Web

Por último, como paso previo al análisis y la discusión de resultados, para facilitar la generación de pacientes virtuales y la clasificación de casos clínicos, hemos desarrollado una página web que permita realizar ambas actividades. Para visualizarla podemos acceder al enlace [https://garridosevillatfg.herokuapp.com/], lo cual nos devolverá el sitio que aparece en la Figura 33.

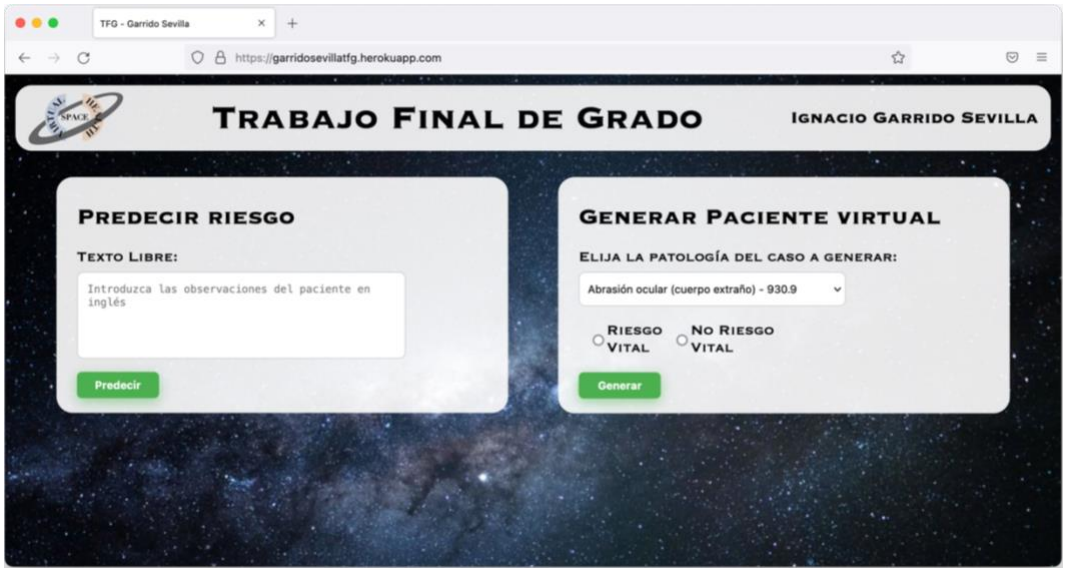

Figura 33 *Página web del TFG*
Fuente: Elaboración propia, accesible en https://garridosevillatfg.herokuapp.com/

Como se puede observar en la página, mediante el panel izquierdo se pueden obtener clasificaciones para cada uno de los modelos generados en el Experimento 6. A su vez, a través del panel derecho se pueden generar pacientes virtuales de cada una de las veinte patologías estudiadas en el proyecto seleccionando si se quiere obtener un caso clínico que presente riesgo vital o no.

En cuanto a su desarrollo, para la parte visual se ha empleado HTML y CSS puro, sin ningún tipo de biblioteca o *framework* para tal propósito. De esta forma, dentro de la carpeta WEB del repositorio asociado al TFG, en el archivo «*templates/index.html*» (Apartado 9.6) se puede encontrar la estructura de la web, mientras que el estilo está definido en el archivo «*static/css/main.css*».

Por otro lado, para introducir las funciones generadas en los anteriores apartados necesitábamos crear una aplicación web basada en Python. Con este propósito, hemos escogido el framework ***Flask*** que permite, con relativamente pocas líneas de código, implementar la funcionalidad deseada. De esta forma, cuando se pulsa en los botones de «Predecir» o «Generar» se accede a un *path* con los atributos introducidos en el que se utilizan las funciones ya descritas para obtener el resultado que se muestra en pantalla. El código de la aplicación se puede encontrar en el archivo «*virtual_patients_app.py*» en el que han sido copiadas las funciones necesarias de los pasos anteriores junto a la funcionalidad requerida para el correcto funcionamiento de los botones. Las líneas de código asociadas a esta última parte se pueden observar en el Apartado 9.7.

Finalmente, una vez desarrollada la página web, para que fuera accesible al público necesitábamos contratar algún servicio de hosting. Para ello hemos creado una cuenta de Heroku[16] que nos permite publicar la web con una url relativamente personalizable y, aunque con algunas limitaciones, de forma totalmente gratuita. Tras confirmar que los recursos ofrecidos eran suficientes para nuestra aplicación, construimos los archivos denominados «*requirements.txt*» en el que se especifican las dependencias del proyecto y «*Procfile*» en el que se indica el comando necesario para ejecutar la aplicación, se sube la estructura de archivos de la web a la plataforma de Heroku y, minutos más tarde, la web es accesible desde el enlace siguiente: [https://garridosevillatfg.herokuapp.com/]

[16] Web oficial: https://www.heroku.com

5 Análisis y discusión de resultados

Una vez explicado en los puntos anteriores el procedimiento seguido, se muestran a continuación los resultados obtenidos tras el entrenamiento y el testeo de los distintos modelos utilizados en cada uno de los experimentos, así como un breve análisis sobre los que elaborar las posteriores conclusiones. Además, cabe destacar que se identifica en **negrita** los modelos con mejor resultado para cada parámetro

Experimento 1

En este caso se usan los 3.000 pacientes virtuales del CIE 692,76, k=all y TF-IDF sin parámetros. El support es 393-357 para las clases 0-1.

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
LinearSVC	0.943	**1.00**	**1.00**	0.973	0.971
Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
MLPerceptron	0.989	0.995	0.994	0.992	0.991

Tabla 5 *Resultados Experimento 1*
Fuente: Elaboración propia

Tras realizar este experimento, se puede observar en la Tabla 5 que la clasificación es realmente buena, destacando el modelo *Multinomial* y el de *Random Forest* sobre el resto, puestos que ambos clasifican de forma perfecta. A su vez, cabe mencionar que, en comparación al resto, el modelo *MLPercetron* obtiene resultados bastante peores en mucho más tiempo mientras que el *Multinomial* es el más rápido de todos.

Experimento 2

En este segundo experimento se utilizan las bases de datos de los Virtual Patients y del 112 de forma separada, sin *lemmatization*, con k=1.000 (siendo el total de términos 2.556) y sin parámetros en la generación de la matriz TF-IDF. A su vez, para la base de datos del 112 se

entrenarán los modelos habiendo sido eliminadas las *stopwords* (como en los Virtual Patiens) y sin haber realizado dicho filtro. En este caso el *support* es de 7.559-7.441 y 18.958-15.082, respectivamente ,para las clases 0-1.

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.948	0.993	0.993	0.971	0.970
LinearSVC	0.920	**0.997**	0.996	0.959	0.957
Random Forest	0.966	**0.997**	**0.997**	**0.982**	**0.981**
MLPerceptron	**0.967**	0.994	0.994	0.980	0.980

Tabla 6 *Resultados Experimento 2 – Virtual Patients*
Fuente: Elaboración propia

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.709	**0.858**	0.799	0.792	0.751
LinearSVC	0.769	0.854	**0.808**	0.816	0.788
Random Forest	0.773	0.834	0.788	0.807	0.780
MLPerceptron	**0.794**	0.837	0.795	**0.818**	**0.794**

Tabla 7 *Resultados Experimento 2 – Casos del 112 – Stopwords eliminadas*
Fuente: Elaboración propia

Como se observa en las Tabla 6, los resultados obtenidos para los pacientes virtuales son claramente superiores a los del 112, sin ser estos últimos malos resultados para una base de datos real. Además, en el primer caso destaca de forma predominante la bondad del modelo *Random Forest*, liderando todas las ratios excepto el *recall* que, prácticamente, lo iguala al mejor caso. Por el contrario, para la segunda base de datos, no hay un claro ganador, aunque en todos los parámetros los modelos se encuentran bastante igualados excepto, de nuevo, el *recall*, donde existe la mayor diferencia entre dos modelos (*Multinomial* y *MLPerceptron*).

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.688	**0.871**	**0.809**	0.790	0.744
LinearSVC	0.772	0.854	0.807	**0.817**	0.789
Random Forest	0.768	0.840	0.793	0.808	0.780
MLPerceptron	**0.788**	0.839	0.796	**0.817**	**0.792**

Tabla 8 *Resultados Experimento 2 – Casos del 112 – Stopwords NO eliminadas*
Fuente: Elaboración propia

Comparando los resultados del modelo habiendo eliminado las *stopwords* y sin llevar a cabo dicho proceso, se observa que no hay un claro ganador puesto que, en algunos casos, se observan mejores resultados en el primero de ellos y otras en el segundo. No obstante, en todos los casos la diferencia es mínima.

Por último, cabe destacar que tanto *Random Forest* como el modelo *MLPerceptron* requieren de una cantidad de tiempo considerablemente mayor a sus dos homólogos, llegando el segundo al máximo de iteraciones establecidas por defecto (200). Sin embargo, aumentar este valor no mejora los datos de forma considerable, aunque sí incrementa en gran medida el tiempo necesario.

Experimento 3

Para este experimento, continuamos en las condiciones anteriores eliminando las *stopwords* en ambas bases de datos y, ahora sí, empleando los parámetros, explicados en el capítulo anterior, para la generación de la matriz TF-IDF.

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.948	**0.999**	**0.999**	0.974	0.973
LinearSVC	0.921	0.996	0.995	0.959	0.957
Random Forest	**0.966**	0.997	0.997	**0.982**	**0.982**
MLPerceptron	0.962	0.966	0.965	0.964	0.964

Tabla 9 *Resultados Experimento 3 – Virtual Patients*
Fuente: Elaboración propia

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.742	**0.856**	**0.803**	0.805	0.771
LinearSVC	0.782	0.843	0.799	**0.816**	**0.790**
Random Forest	0.774	0.840	0.794	0.811	0.784
MLPerceptron	**0.791**	0.815	0.773	0.804	0.781

Tabla 10 *Resultados Experimento 3 – Casos del 112*
Fuente: Elaboración propia

De nuevo, observamos que en ninguna de las dos bases de datos existe un claro ganador pues ningún modelo destaca sobre el resto de forma significativa. No obstante cabe mencionar los buenos resultados del modelo *Multinomial* respecto al resto, siendo éste el más sencillo y, con diferencia, el más rápido en tiempo de ejecución. A su vez se observa de nuevo la diferencia a nivel general entre los resultados de ambas bases de datos, siendo claramente superiores los de los pacientes virtuales.

Con respecto al experimento anterior, se aprecia de forma general una ligera mejoría, siendo esta más destacable en los valores de *recall*, aunque no se puede decir que sea verdaderamente significativa.

Por último, a tenor de los parámetros introducidos en la generación de la matriz TF-IDF cabe mencionar que el uso de monogramas (1,1) o bigramas (1,2) no presenta una mejoría clara, por lo que, de cara a ulteriores experimentos, mantendremos la opción de utilizar el que presente mejores resultados.

Experimento 4

En este experimento se continúan manteniendo las condiciones del anterior, pero añadiendo el proceso de *lemmatization* al texto libre de los casos.

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.938	**0.998**	**0.997**	0.968	0.967
LinearSVC	0.917	0.994	0.993	0.956	0.954
Random Forest	0.962	0.993	0.993	**0.978**	**0.977**
MLPerceptron	**0.964**	0.966	0.965	0.965	0.965

Tabla 11 *Resultados Experimento 4 – Virtual Patients*
Fuente: Elaboración propia

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.744	**0.855**	**0.803**	0.806	0.772
LinearSVC	0.781	0.843	0.798	**0.816**	**0.790**
Random Forest	0.777	0.842	0.797	0.813	0.787
MLPerceptron	**0.782**	0.821	0.776	0.803	0.779

Tabla 12 *Resultados Experimento 4 – Casos del 112*
Fuente: Elaboración propia

Como se observa en las Tabla 11 con respecto a las del experimento anterior, no se aprecia una mejoría significativa en ningún valor. A su vez, el modelo *Multinomial* continúa destacando sobre el resto en las ratios de *specifity* y *precision*, aunque, en general, todos los modelos se encuentran bastante igualados.

Experimento 5

En este último experimento, se empleará la técnica de lemmatization, la utilización de parámetros en la generación de la matriz TF-IDF, el conjunto de términos resultantes del paso anterior y el filtrado de *stopwords*. Sin embargo, en este experimento se introducen dos elementos diferenciadores:

- **Unificación**: En este caso, en lugar de realizar los experimentos de forma separada, ambas bases de datos se fusionarán en una sola. Gracias a que ambas cuentan con una estructura homogénea, este paso es tan sencillo como unir ambos conjuntos de datos en un único *dataframe* de 196.160 casos. A su vez, ,mantendremos el porcentaje 75-25% para *train-test*.

- **Ensemble**: Por otro lado, como ya ha sido mencionado previamente en el trabajo, al utilizar varios modelos en la clasificación o predicción de datos, se puede utilizar una agrupación de los mismos para obtener un modelo mejor gracias a la compensación de errores entre ellos. Con dicho propósito, se ha utilizado el objeto *VotingClassifier* de la librería Scikit-learn, la cual permite aglutinar distintos modelos en uno sólo. Debido a que todos ellos ofrecen un resultado cualitativo, el valor final se decidirá por votación mayoritaria. Sin embargo, al contar con un número par de clasificadores, el caso de empate es bastante probable y para estas circunstancias ha sido añadido el parámetro *voting=hard*, que define la asignación con la etiqueta de menor valor (en este caso 0, es decir, no riesgo) en los casos en los que no exista mayoría clara y que, para nosotros, supone el peor caso: clasificar como de NO RIESGO un caso de RIESGO.

MODELO	*RECALL*	*SPEC*	*PREC*	*ACC*	*F1*
MultinomialNB	0.796	0.888	0.859	0.846	0.826
LinearSVC	0.808	0.889	0.861	0.852	0.834
Random Forest	0.826	0.885	0.859	0.858	0.842
MLPerceptron	**0.848**	0.871	0.849	0.860	**0.848**
Ensemble	0.820	**0.896**	**0.870**	**0.861**	0.844

Tabla 13 *Resultados Experimento 5 – Bases de datos unificada*
Fuente: Elaboración propia

Como se observa en la Tabla 13, en este caso existe un mismo vencedor en tres de los cinco parámetros a analizar, que es el *Ensemble*. Sin embargo, cabe mencionar que el tiempo de ejecución es muy superior al del resto de modelos y que, a pesar de sus mejores resultados, no presentan una diferencia realmente significativa.

Así mismo, el modelo *MLPerceptrón* sigue destacando como el que presenta mejor resultado para la ratio *recall* y, en este caso, también en F1.

6 Conclusiones

En el presente proyecto ha sido abordada la tarea de generar un clasificador capaz de determinar la existencia de riesgo vital en un caso clínico durante la exploración espacial mediante el análisis de un texto que describa la situación y sintomatología de un paciente. Para ello, ha sido necesario llevar a cabo distintas actividades como la elaboración de un *web scraper* que permitiera automatizar la recogida de datos e información para un conjunto de patologías escogido. A su vez, se ha tenido que concretar dicho grupo de condiciones clínicas para centrar el trabajo dentro del marco de los viajes espaciales.

Además, se han llevado a cabo tareas de procesado de textos, así como el diseño de un generador de pacientes virtuales que permitiera trabajar de forma indistinta entre casos reales y ficticios. Por último, utilizando casos asociados a las emergencias del 112 y los mencionados pacientes artificiales, se han podido llevar a cabo distintos modelos con los que obtener clasificadores capaces de identificar situaciones de riesgo.

Una vez realizado todo el proceso descrito en el Capítulo anterior incluida una serie de experimentos, procedemos a extraer algunas conclusiones interesantes, a nivel general, que merecen ser destacadas:

- Las técnicas de web scraping son tremendamente útiles para la obtención de datos de forma masiva y automática. Sin embargo, para que su resultado sea satisfactorio o bien que su uso aporte más beneficios que complicación, es necesario partir de datos que se encuentren bien estructurados, lo cual no siempre es fácil o incluso posible. Por ello, el acceso a bases de datos de calidad se antoja como un punto trascendental dentro de este tipo de trabajos

- Aunque, como se comentará a continuación, los resultados de la generación de pacientes virtuales han sido relativamente satisfactorios, el uso de datos del mundo real sigue siendo ineludible. El motivo de este hecho es que simular de forma fidedigna la variabilidad presente en los informes médicos es realmente complejo, por lo que aunque esta primera aproximación cumple en gran medida las expectativas depositadas, precisa ser complementado con datos reales.

- Como ha sido comentado anteriormente, la investigación dentro del campo médico orientada a la exploración del espacio puede suponer, no solo un incremento en la seguridad de los astronautas de cara a los futuros viajes espaciales, sino que también podría mejorar la calidad de vida de las personas en la Tierra gracias a la simbiosis entre tecnología y salud. Además, las previsiones de la actividad aeroespacial, que auguran un futuro con enorme potencial para este sector, unido a las enormes fuentes

de financiación que se espera reciba dicha industria, hace que la posibilidad de realizar investigaciones en este campo pueda ser una gran oportunidad.

Una vez extraídas estas conclusiones generales, procedemos a comentar aquellos aspectos más relevantes que se observan en los resultados de los experimentos llevados a cabo.

- El primer hecho a reseñar hace referencia a los magníficos resultados obtenidos, de forma general, por todos los modelos cuando el conjunto de datos de entrada se componía de forma exclusiva por pacientes virtuales. Esto se debe, en gran medida, al mecanismo de generación de los pacientes virtuales, ya que la existencia de riesgo vital se identifica con la presencia, dentro del conjunto de palabras, de alguna de las catalogadas como de riesgo. Es decir, a pesar de la aparente complejidad en la generación de los casos, la identificación del riesgo se puede llevar a cabo con una simple condición lógica del tipo «SI alguna de estas palabras coincide con alguna de estas otras, ENTONCES presenta riesgo», la cual es fácilmente identificada por los modelos. De esta forma, mientras que los casos reales es probable que sean menos ricos en información que los pacientes virtuales y, por tanto, sean más difíciles de predecir, los segundos están altamente informados y no representan de forma fidedigna las condiciones del mundo real.

- Cabe destacar que de los cuatro modelos el *Naive Bayes* o *Multinomial*, es con diferencia, el más sencillo de todos. Esta simpleza alcanza su máxima expresión en los tiempos de ejecución puesto que, en todos los casos, es el más rápido con gran diferencia. Sin embargo, esta velocidad no le penaliza de forma significativa con el resto de los modelos, por lo que se puede decir que es el que presenta mejor relación entre resultados y consumo de recursos.

- Respecto al uso de *stopwords* para llevar a cavo el filtrado de los datos de entrada se ha podido comprobar que no supone una mejora significativa en los resultados. En este caso, realizar dicho filtrado no supone un coste computacional elevado, sin embargo, de cara a proyectos con cantidades de datos mucho mayores, habría que estudiar a fondo la conveniencia de utilizar dicha práctica.

- De igual forma, el uso de la práctica de *lemmatization* como vemos en el Experimento 4 mejora y empeora algunos parámetros en todos los modelos, aunque en ningún caso de forma relevante. Por este motivo, se puede extraer la misma conclusión del punto anterior y, de cara a futuros proyectos, debería ser analizada con cautela la verdadera utilidad de este procedimiento.

- En cuanto al uso de parámetros en la generación de la matriz TF-IDF se puede observar que tampoco presenta una mejora significativa el uso de los 1.000 términos con mayor peso ni los parámetros para filtrar las palabras que aparecen en un mínimo/máximo número de textos. No obstante, cabe destacar que en el modelo *Multinomial* todos los valores se igualan o mejoran para la base de datos de virtual patiens y para la del 112 aunque en este caso, el specificity disminuye levemente, aunque es probable que estas diferencias se deban al sesgo de la muestra y no a una mejora intrínseca del uso de los mismos. A su vez, el uso de unigramas o bigramas en dicha matriz supone una mejora según el caso por lo que no está claro si alguna de las opciones es claramente mejor a la otra.

- Respecto al uso conjunto de ambas bases de datos, se puede apreciar de forma clara cómo los resultados se encuentran dentro del rango de los obtenidos previamente utilizando los conjuntos de datos de forma separada. En cualquier caso, el uso de pacientes virtuales se muestra totalmente compatible con el de casos reales, por lo que, aunque el proceso de generación de pacientes con riesgo necesite una complejidad mayor para acercarse a la realidad, su uso puede suponer una gran oportunidad de cara a este tipo de proyectos.

- Como se ha comentado previamente, las características propias de nuestro experimento hacen que se deba valorar de forma más positiva aquellos modelos cuyo resultado en el parámetro *Recall* sea superior. Esto nos asegura que el número de falsos negativos sea lo más bajo posible que es precisamente lo que se pretende conseguir entre otros objetivos. De esta forma, analizando los resultados de todos los experimentos se puede observar que el modelo *MLPerceptron* destaca en la mayoría de los casos para esta ratio.

- En cuanto al uso del *Ensemble*, se observa que destaca en tres de los cinco parámetros utilizados. De este modo se confirma su utilidad de cara a compensar los errores introducidos por distintos modelos, aunque sería interesante utilizarlo cuando el número de modelos fuera impar, de forma que no existiera ningún sesgo para ninguna de las clases.

- Por último, respecto al experimento final, se observa que prácticamente en todos los parámetros calculados para los cinco modelos, se supera el valor de 80% que se pretendía alcanzar como uno de los objetivos iniciales. De esta forma, se confirma la bondad y capacidad de todos los modelos empleados para clasificar el riesgo en pacientes.

Tras haber obtenido las anteriores conclusiones se sientan las bases de cara a las líneas futuras del proyecto. En primer lugar, la práctica de los distintos experimentos abordados ha permitido obtener conclusiones interesantes que pueden resultar realmente útiles de cara a ulteriores ensayos. La criba realizada a partir de algunos resultados, no totalmente satisfactorios, sirve de guía para seguir avanzando en ulteriores investigaciones. En segundo lugar, la definición realizada del algoritmo para la generación de pacientes virtuales se presenta como la primera iteración de lo que podría llegar a ser un generador más complejo y sofisticado. Por último, el desarrollo de la página web nos muestra las enormes posibilidades de simplificar, en una pantalla con una interfaz intuitiva y con un único «click», una tarea tan compleja como el cuidado médico y el diagnóstico (de la presencia de riesgo vital en este caso).

Por todo ello consideramos este trabajo una buena aproximación inicial tanto a la investigación médica enfocada al espacio como a lo que podría ser, en el futuro, el clasificador de un sistema completo y avanzado como MEDEA.

7 Líneas futuras

El trabajo presentado en este documento sienta las bases previas del clasificador de riesgos que sería usado por el sistema MEDEA, el cual ha sido comentado anteriormente. No obstante, la consecución del sistema completo es una tarea compleja y con altas necesidades tanto técnicas como temporales. Por este motivo, se ha elaborado un posible plan de actuación, dividido en cuatro fases, para la realización del proyecto completo.

	FASE 1	FASE 2	FASE 3	FASE 4
ENTRADA	**Licencia + Patente + TRL 3**	**TRL 4**	**TRL 6**	**TRL 8**
SALIDA	**TRL 4**	**TRL 6**	**TRL 8**	**TRL 9**
DURACIÓN	**2 años**	**12 meses**	**De 1 a 2 años**	**De 6 a 12 meses**
OBJETIVOS	**Desarrollo de módulos. Testeo y primera iteración de Sistema Autónomo y Aprendizaje Adaptativo.**	**Validación de uso de módulos críticos con datos reales. Acuerdos con organismos espaciales (NASA, ESA).**	**Desarrollo completo de MEDEA. Integración propia y con otros sistemas de la nave espacial.**	**Test global de funcionamiento. Verificación de funcionamiento a nivel comercial y escalabilidad.**

Tabla 14 *Fases del desarrollo técnico de MEDEA*
Fuente: Elaboración propia

Por cada una de ellas, ha sido identificado el estado inicial y final mediante los niveles de *Technology Readiness Level* [27], en adelante TRL, concepto que hace referencia al nivel de madurez de una tecnología. Aunque surge de la NASA en los años 70 para el ámbito de la aeronáutica o espacial, actualmente se ha generalizado a casi cualquier tipo de proyecto tecnológico. Gracias a su sistema de nueve niveles, como vemos en la Figura 34, puede ser utilizado para evaluar desde la concepción inicial de un producto hasta su comercialización.

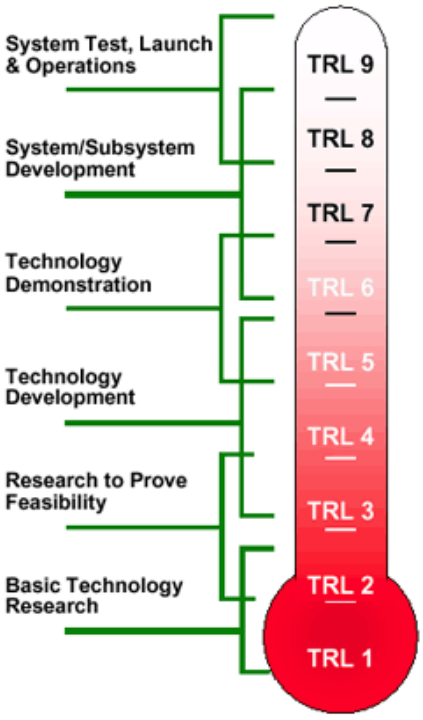

Figura 34 Niveles TRL
Fuente: [27]

Al mismo tiempo, resulta conveniente realizar una agrupación de los distintos niveles que permitan, de forma genérica, obtener una perspectiva global de dónde se sitúa cada nivel TLR. Como se observa en la Figura 34, se pueden encasillar, a grandes rangos, en tres grupos:

- El primero y menos maduro (TRL 1 - 4). En él destaca el trabajo en un entorno de laboratorio y la investigación de nuevas y disruptivas tecnologías. Predomina el trabajo del marco teórico, así como las pruebas de concepto y la validación en entornos específicos y controlados.

- El segundo, como grupo intermedio, destaca por el desarrollo tecnológico sobre los resultados del grupo anterior. Para ello se generan simulaciones a partir de datos reales y se realizan los primeros prototipos.

- El tercer y último grupo, es el que presenta un mayor nivel de madurez. Las validaciones controladas y el desarrollo dejan paso a la verificación en entornos reales y se comienza a definir el despliegue final de la tecnología empleada.

NIVEL TRL	SUBGRUPO
TRL 9	Entorno real Innovación
TRL 8	
TRL 7	
TRL 6	Entorno de simulación Desarrollo
TRL 5	
TRL 4	Entorno de laboratorio Investigación
TRL 3	
TRL 2	
TRL 1	

Tabla 15 *Subgrupos de TRLs*
Fuente: Elaboración propia

Fase 1

La primera fase del desarrollo tecnológico parte de un escenario base TRL 3, es decir, la tecnología a desplegar ya cuenta con una investigación previa y pruebas de concepto válidas, en forma de patente, que permiten verificar la tecnología como candidata a ser explotada de forma útil. Una primera aproximación a esta posible patente sería el presente Trabajo de Fin de Grado.

A su vez, los **objetivos** de esta fase serían:

- Recopilar información y datos suficientes para un correcto entrenamiento de los modelos de datos a desarrollar. Reforzar la investigación médica de las patologías a tratar.
- Trabajar con especial dedicación en el proceso de análisis de datos. En esta fase del proyecto es indispensable definir una buena base para posteriores iteraciones de los mismos.
- Primera iteración del software. Desarrollo básico de los módulos críticos de MEDEA, es decir el sistema autónomo y el sistema de aprendizaje adaptativo.
- Realizar el testeo de las partes desarrolladas y la documentación de los procesos seguidos, así como del software y modelos trabajados.

Fase 2

La segunda fase del proyecto arranca al final de la anterior, partiendo de un nivel TRL 4 y con la perspectiva de comenzar a dejar la visión de laboratorio atrás y centrarse en la simulación, el uso de datos reales para la verificación y el desarrollo como tal.

El eje central de este período será tanto verificar el funcionamiento de los módulos críticos desarrollados en la fase anterior como realizar los ajustes necesarios para su correcto funcionamiento y optimización. De esta forma, los objetivos de esta fase serían los siguientes:

- Iterar sobre las primeras versiones de los módulos críticos desarrollados anteriormente. Aplicar los cambios necesarios y valorar posibles mejoras mediante la realización de prototipos.

- Diseño de evaluaciones y sistemas de verificación a partir de datos reales y demostración del funcionamiento de los módulos con datos reales.

- Búsqueda de acuerdos para la obtención de datos con instituciones relevantes dentro del sector (por ejemplo, la NASA o la ESA para el uso de datos procedentes de la ISS).

Fase 3

Esta penúltima fase se adentra de lleno en el entorno real y la certificación del sistema en base a su uso esperado final. Sin embargo, es conveniente resaltar que para superar el TRL 6 se debe contar con un desarrollo completo del sistema.

Por tanto, la primera parte de esta fase corresponderá al desarrollo de todo el sistema MEDEA en su conjunto, es decir, añadir el módulo de interoperabilidad y del soporte ético-legal a los ya preexistentes. Tras ello, el sistema debe ser validado de forma individual para poder comenzar con la integración del mismo. Una vez el sistema se encuentra funcionando en su conjunto, habremos alcanzado el TRL 7 y comenzarán las demostraciones en entornos reales.

Por todo ello, los objetivos de este período son:

- Desarrollo completo del sistema MEDEA. Implantación de los módulos pendientes del sistema y validación interna.

- Integración de todos los módulos entre ellos, así como con otros sistemas presentes, o que se espera que lo estén, en una nave espacial (*v.g.,* historia clínica electrónica, soporte vital, …).

- Refinamientos finales y certificación del sistema completo mediante pruebas y demostraciones en entornos reales.

Fase 4

Tras alcanzar el TRL 8, la cuarta y última fase de desarrollo estará marcada por las pruebas finales en entorno real y la verificación de su funcionamiento óptimo en el escenario de trabajo para el que ha sido creado.

Esta fase, que busca alcanzar el TRL 9, se caracteriza por ser la antesala del producto final. Por tanto, esta fase debería concentrar los esfuerzos en las pruebas en entornos reales y dotar de publicidad al proyecto.

Tras todo ello, se habrá alcanzado el nivel de madurez suficiente como para situarse en el TRL 9 y, en consecuencia, ser considerada como candidata a comercializarse en el mercado.

8 Bibliografía

[1] E. Romero y D. Francisco, «The NASA human system risk mitigation process for space exploration,» *Acta Astronautica,* vol. 175, pp. 606-615, 2020.

[2] Mckinsey & Company, *An executive's guide to AI,* https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/an-executives-guide-to-ai, 2020.

[3] M. Raza y P. Cinquegrana , *Deep Learning: The Latest Trend in AI and ML,* https://www.qubole.com/blog/deep-learning-the-latest-trend-in-ai-and-ml/, 2021.

[4] D. Varghese , *Comparative Study on Classic Machine learning Algorithms,* https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222, 2018.

[5] A. Sharma, *Decision Tree vs. Random Forest – Which Algorithm Should you Use?,* https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/, 2020.

[6] J. M. Heras, *Random Forest (Bosque Aleatorio): combinando árboles,* https://www.iartificial.net/random-forest-bosque-aleatorio/, 2020.

[7] E. Akgün y M. Demir, *Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks,* https://doi.org/10.21449/ijate.444073, 2018.

[8] Scikit Learn - Documentation, *Neural network models (supervised),* https://scikit-learn.org/stable/modules/neural_networks_supervised.html, 2020.

[9] D. Calvo, *Definición de red neuronal artificial,* https://www.diegocalvo.es/definicion-de-red-neuronal/, 2017.

[10] E. A. Zanaty, *Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification,* https://doi.org/10.1016/j.eij.2012.08.002, 2012.

[11] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features,* DOI:10.17877/DE290R-5097, 1998.

[12] E. Kim, *Everything You Wanted to Know about the Kernel Trick,* http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html, 2017.

[13] J. M. Heras, *Ensembles: voting, bagging, boosting, stacking,* https://www.iartificial.net/ensembles-voting-bagging-boosting-stacking/, 2019.

[14] R. Artola, La carrera espacial: Del Sputnik al Apollo 11, Madrid, España: Alianza Editorial, 2019.

[15] NASA, «As Artemis Moves Forward, NASA Picks SpaceX to Land Next Americans on Moon,» 2021. [En línea]. Available: https://www.nasa.gov/press-release/as-artemis-moves-forward-nasa-picks-spacex-to-land-next-americans-on-moon .

[16] NASA, «Human Exploration of Mars Design Reference Architecture 5.0,» 2009. [En línea]. Available: https://www.nasa.gov/pdf/373667main_NASA-SP-2009-566-ADD.pdf.

[17] J. Avila-Tomás, M. A. Mayer-Pujadas y V. J. Quesada-Varela, *La inteligencia artificial y sus aplicaciones en medicina I: introducción antecedentes a la IA y robótica,* vol. 52, DOI: 10.1016/j.aprim.2020.04.013, 2020, pp. 778-784.

[18] A. Kumar, M. Ramachandran, A. H. Gandomi, R. Patan, S. Lukasik y R. K. Soundarapandian, *A deep neural network based classifier for brain tumor diagnosis,* Ravichandran Kattur , 2019.

[19] M. A. Douglass, J. P. Casale, J. A. Skirvin y M. V. DiVall, «A Virtual Patient Software Program to Improve Pharmacy Student Learning in a Comprehensive Disease Management Course,» 2013. [En línea]. Available: DOI: 10.5688/ajpe778172.

[20] A. H. Pollack, T. D. Simon, J. Snyder y W. Pratt, *Creating synthetic patient data to support the design and evaluation of novel health information technology,* doi.org/10.1016/j.jbi.2019.103201, 2019.

[21] J. M. Garcia-Gomez, *Basic principles and concept design of a real-time clinical decision support system for autonomous medical care on missions to Mars based on adaptive deep learning,* https://arxiv.org/submit/3392580, 2020.

[22] L. H. Stewart, D. Trunkey y G. S. Rebagliati, *Emergency medicine in space,* The Journal of Emergency Medicine: doi:10.1016/j.jemermed.2006.05.031.

[23] A. Rajkomar, J. Dean y I. Kohane, *Machine Learning in Medicine,* New England Journal of Medicine: doi:10.1056/NEJMra1814259, 2019.

[24] E. Matthes, Python Crash Course, San Francisco: No Starch Press, 2016.

[25] D. Sarkar y R. Bali, *The Python Machine Learning Ecosystem,* DOI:10.1007/978-1-4842-3207-1_2, Ed., 2017, pp. 67-118.

[26] G. T. A. Kovacs y M. Shadden, *Analysis of age as a factor in NASA astronaut selection and career landmarks,* doi.org/10.1371/journal.pone.0181381, 2017.

[27] ESA, «Technology readiness levels handbook for space applications,» 2008. [En línea]. Available: https://artes.esa.int/sites/default/files/TRL_Handbook.pdf.

[28] US News & University of Pennsylvania, «Best Countries 2020,» 2020. [En línea]. Available:

https://media.beam.usnews.com/8e/b0/c99b324c4a0a8c1f6dd7c76d903c/200108-best-countries-overall-rankings-2020.pdf.

[29] ESA, «Space19+,» 2019. [En línea]. Available: https://blogs.esa.int/space19plus/es/.

[30] UK Gov., «UK involvement in the EU Space Programme,» 2020. [En línea]. Available: https://www.gov.uk/guidance/uk-involvement-in-the-eu-space-programme.

[31] Comisión Europea, «The Digital Economy and Society Index (DESI),» 2020. [En línea]. Available: https://ec.europa.eu/digital-single-market/en/digital-economy-and-society-index-desi.

[32] J. Henry, Y. Pylypchuk, T. Searcy y V. Patel, «Adoption of Electronic Health Record Systems among U.S. Non-Federal Acute Care Hospitals: 2008-2015,» 2015. [En línea]. Available: https://www.healthit.gov/sites/default/files/briefs/2015_hospital_adoption_db_v17.pdf.

[33] OECD, «The Space Economy in Figures: How Space Contributes to the Global Economy,» 2019. [En línea]. Available: https://doi.org/10.1787/c5996201-en.

[34] P. Calla, D. Fries y C. Welch, «Asteroid mining with small spacecraft and its economic feasibility,» 2019. [En línea]. Available: https://arxiv.org/pdf/1808.05099.pdf.

[35] P. K. Singh y R. S. Singh, «Environmental and social impacts of mining and their mitigation,» 2016. [En línea]. Available: https://www.researchgate.net/publication/308937912_Environmental_and_social_impacts_of_mining_and_their_mitigation.

[36] R. Baldridge y B. Curry, «What is a startup?,» 2021. [En línea]. Available: https://www.forbes.com/advisor/investing/what-is-a-startup/.

[37] M. E. Graebner, «Momentum and serendipity: how acquired leaders create value in the integration of technology firms,» 2004. [En línea]. Available: https://doi.org/10.1002/smj.419.

[38] A. Das Gupta, Strategic Human Resource Management: formulating and implementing, New York: Routledge, 2020.

[39] Comisión Europea, «Directrices éticas para una IA fiable,» 2018. [En línea]. Available: https://op.europa.eu/es/publication-detail/-/publication/d3988569-0434-11ea-8c1f-01aa75ed71a1.

[40] Z. S. Patel, T. J. Brunstetter, W. J. Tarver, A. M. Whitmire, S. R. Zwart, S. M. Smith y J. L. Huff, «Red risks for a journey to the red planet: The highest priority human health risks for a mission to Mars.,» 2020. [En línea]. Available: https://doi.org/10.1038/s41526-020-00124-6.

9 Anexos

9.1 Anexo I: Archivo «00_CIE_emedicine_info.xlsx»

Nº NASA	MEDICAL CONDITION	CIE	WEB
1	Abdominal Injury	959,12	https://emedicine.medscape.com/article/1984639-overview
2	Abdominal Wall Hernia	553,29	https://emedicine.medscape.com/article/189563-overview
3	Abnormal Uterine Bleeding	626,8	https://emedicine.medscape.com/article/795587-overview
4	Acute Arthritis	716,9	https://emedicine.medscape.com/article/335692-overview
5	Acute Cholecystitis/Biliary Colic	575	https://emedicine.medscape.com/article/171886-overview
6	Acute Compartment Syndrome	958,9	https://emedicine.medscape.com/article/307668-overview
7	Acute Diverticulitis	562,11	https://emedicine.medscape.com/article/173388-overview
8	Acute Glaucoma	365,22	https://emedicine.medscape.com/article/1206956-overview
9	Acute Pancreatitis	577	https://emedicine.medscape.com/article/181364-overview
	Acute pericarditis	420,9	https://emedicine.medscape.com/article/156951-overview
10	Acute Prostatitis	601	https://emedicine.medscape.com/article/785418-overview
11	Acute Radiation Syndrome	990	https://emedicine.medscape.com/article/1979331-overview
12	Acute Sinusitis	461,9	https://emedicine.medscape.com/article/232670-overview
13	Aerobic Capacity Loss	E009,2	
14	Allergic Reaction (mild tomoderate)	995,3	https://emedicine.medscape.com/article/135959-overview
15	Altitude Sickness/Hypoxia	993,2	https://emedicine.medscape.com/article/768478-overview
17	Anaphylaxis	995,6	https://emedicine.medscape.com/article/135065-overview
16	Angina/Myocardial Infarction	411,1	https://emedicine.medscape.com/article/155919-overview
18	Ankle Sprain/Strain	845	https://emedicine.medscape.com/article/1907229-overview
19	Anxiety	300	https://emedicine.medscape.com/article/286227-overview
20	Appendicitis	540,9	https://emedicine.medscape.com/article/773895-overview
21	Atrial Fibrillation/Flutter	427,31	https://emedicine.medscape.com/article/151066-overview
22	Back Injury Code 959.19	724,5	https://emedicine.medscape.com/article/95444-overview
23	Back Pain (SAS)	724,5	https://emedicine.medscape.com/article/822462-overview
24	Barotrauma (sinus block)	993,1	https://emedicine.medscape.com/article/768618-overview
25	Behavioral Emergency	301	https://emedicine.medscape.com/article/287913-overview
26	Burns (secondary to fire)	949	
27	Cardiogenic Shock (secondary to infarction)	785,51	https://emedicine.medscape.com/article/152191-overview
28	Chest Injury	959,11	https://emedicine.medscape.com/article/428723-overview
29	Choking/Obstructed Airway	799,01	https://emedicine.medscape.com/article/1988699-overview
30	CO2 Headache/ICP/Cognitive	986	https://emedicine.medscape.com/article/819987-overview
31	Constipation (SAS)	564	https://emedicine.medscape.com/article/184704-overview
32	Decompression Sickness (secondary to EVA)	993,3	https://emedicine.medscape.com/article/769717-overview
	Deep vein thrombosis	453,4	https://emedicine.medscape.com/article/1911303-overview
33	Dental: Exposed Pulp, Caries, Abscess, Tooth Filling/Crown Loss	525,9	https://emedicine.medscape.com/article/763538-overview
34	Depression	311	https://emedicine.medscape.com/article/286759-overview
35	Diarrhea	787,91	https://emedicine.medscape.com/article/928598-overview
36	Dust Exposure (celestial)	504	
37	Elbow Dislocation	832	https://emedicine.medscape.com/article/96758-overview
38	Elbow Sprain/Strain	841,9	https://emedicine.medscape.com/article/1230902-overview
39	Electric Shock Injury	994,8	https://emedicine.medscape.com/article/770179-overview
40	Eye Abrasion (foreign body)	930,9	https://emedicine.medscape.com/article/1195402-overview
41	Eye Chemical Burn	940	https://emedicine.medscape.com/article/798696-overview
42	Eye Corneal Ulcer	370	https://emedicine.medscape.com/article/1195680-overview
43	Eye Infection	373,9	https://emedicine.medscape.com/article/1191730-overview
44	Eye Penetration (foreign body)	871,7	https://emedicine.medscape.com/article/1195581-overview
45	Finger Dislocation	834	https://emedicine.medscape.com/article/823676-overview
46	Fingernail Delamination (EVA)	703,9	
47	Gastroenteritis	558,1	https://emedicine.medscape.com/article/176515-overview
	Gastrointestinal bleeding	578,9	https://emedicine.medscape.com/article/187857-overview
48	Head Injury	959,01	https://emedicine.medscape.com/article/1163653-overview
49	Headache (late SAS)	784	https://emedicine.medscape.com/article/1142556-overview
50	Hearing Loss	389,9	https://emedicine.medscape.com/article/856313-overview

Nº NASA	MEDICAL CONDITION	CIE	WEB
51	Hemorrhoids	455,6	https://emedicine.medscape.com/article/775407-overview
52	Herpes Zoster	53,9	https://emedicine.medscape.com/article/1132465-overview
53	Hip Sprain/Strain	843,9	https://emedicine.medscape.com/article/87043-overview
54	Hip/Proximal Femur Fracture	820	https://emedicine.medscape.com/article/824856-overview
55	Hypertension	401,9	https://emedicine.medscape.com/article/241381-overview
	Hypoglicemia	251,2	https://emedicine.medscape.com/article/122122-overview
56	Immune System Dysfunction/Illness	279,9	
	Increased Intracranial Pressure		https://emedicine.medscape.com/article/1214410-overview
57	Indigestion	536,8	
58	Influenza	487	https://emedicine.medscape.com/article/219557-overview
59	Insomnia	780,52	https://emedicine.medscape.com/article/1187829-overview
60	Knee Sprain/Strain	844,9	https://emedicine.medscape.com/article/826792-clinical
61	Landing Loads/Injuries	845	
62	Lower Extremity Stress Fracture	733,93	https://emedicine.medscape.com/article/86808-overview
63	Lumbar Spine Fracture	805	https://emedicine.medscape.com/article/1264191-overview
64	Malnutrition	263	https://emedicine.medscape.com/article/985140-overview
66	Medication Overdose/Reaction	977,9	
65	Microbial-Host Interaction	8,8	
67	Mouth Ulcer	528,2	https://emedicine.medscape.com/article/867080-overview
68	Muscle Atrophy	728,2	https://emedicine.medscape.com/article/1170572-overview
69	Nasal Congestion (SAS)	478,19	https://emedicine.medscape.com/article/134825-overview
70	Nephrolithiasis (renal stone)	592	https://emedicine.medscape.com/article/437096-overview
71	Neurogenic Shock	308,9	
72	Nose Bleed (SAS)	784,7	https://emedicine.medscape.com/article/863220-overview
73	Orthostatic Intolerance	458	https://emedicine.medscape.com/article/902155-overview
74	Otitis Media/Externa	382,9	https://emedicine.medscape.com/article/994550-overview
75	Paresthesia	782	
76	Pharyngitis	462	https://emedicine.medscape.com/article/764304-overview
	Pneumonia	486	https://emedicine.medscape.com/article/300455-overview
	Pulmonary embolism	415,1	https://emedicine.medscape.com/article/300901-overview
77	Respiratory Infection	465,9	https://emedicine.medscape.com/article/302460-overview
78	Retinal Detachment	361	https://emedicine.medscape.com/article/798501-overview
98	SANS/VIIP – Spaceflight Associated Neuro-Ocular Syndrome/Visual Impairment –	378,9	
79	Seizures/Other convulsions	345	
80	Sepsis	995,91	https://emedicine.medscape.com/article/234587-overview
81	Shoulder Dislocation	831	https://emedicine.medscape.com/article/93323-overview
82	Shoulder Sprain/Strain (EVA)	840,9	https://emedicine.medscape.com/article/828642-overview
83	Skin Abrasion/Laceration	919	
84	Skin Infection	686,9	https://emedicine.medscape.com/article/1830144-overview
85	Skin Rash	782,1	https://emedicine.medscape.com/article/801222-overview
86	Small Bowel Obstruction	560,9	https://emedicine.medscape.com/article/774140-overview
87	Smoke Inhalation	987,9	https://emedicine.medscape.com/article/771194-overview
88	Space Adaptation Sickness (SAS)/Neurovestibular	386,12	
89	Stroke	430-438	https://emedicine.medscape.com/article/1916852-overview
91	Sudden Cardiac Arrest	427,5	https://emedicine.medscape.com/article/151907-overview
90	Sunlight Exposure/Sunburn	692,76	https://emedicine.medscape.com/article/773203-overview
92	Toxic Exposure (ammonia, etc.)		https://emedicine.medscape.com/article/820298-overview
93	Traumatic Hypovolemic Shock	958,4	https://emedicine.medscape.com/article/760145-overview
94	Urinary Incontinence (SAS)	788,3	https://emedicine.medscape.com/article/452289-overview
95	Urinary Retention (SAS)	788,2	
96	Urinary Tract Infection	599	https://emedicine.medscape.com/article/231574-overview
97	Vaginal Yeast Infection	112,1	https://emedicine.medscape.com/article/233101-overview
99	Wrist Fracture	814,09	https://emedicine.medscape.com/article/828746-overview
100	Wrist Strain/Sprain	842	https://emedicine.medscape.com/article/98552-overview

9.2 Anexo II: Archivo «_01_scrape_emedicine.ipynb»

Imports

```
import pandas as pd
from bs4 import BeautifulSoup
import re
import requests
import time
import random
```

CONFIGURATION

Get data

```
webs = pd.read_excel("data/00_CIE_emedicine.xlsx")
```

FUNCTIONS

```
def trim_text(text):
    return re.sub("\n\n+", "\n\n", text).strip()

def get_page(soup,content):
    try:
        text =soup.find(id=content)
            .find(class_='refsection_content').text
        return trim_text(text)
    except:
        try:
            if content == 'content_b2':
                div_b1 = soup.find(id='content_b1')
                text = div_b1.find_next_sibling("div")
                   .find(class_='refsection_content').text
                return trim_text(text)
        except:
            return ""
        return ""

def get_online_info(row):
    try:
        web = row["WEB"]
        r = requests.get(web.replace('overview','clinical'))
        soup = BeautifulSoup(r.text, 'lxml')
        time.sleep(random.randint(5,15))
        return [
            get_page(soup,'content_b1'),
            get_page(soup,'content_b2')
        ]
    except:
        return ["",""]

def get_offline_info(row):
    try:
        cie = str(row["CIE"])
        name = "offline_data/" + cie.replace('.',',') + ".txt"
        f = open(name, "r")
        return f.read()
```

```
    except:
        return ""

def online_down(info):
    return (info[0] == "") & (info[1] == "")
```

GET ONLINE/OFFLINE DATA

```
history = []
examination = []
global_info = []
for index, row in webs.iterrows():
    info = get_online_info(row)
    history.append(info[0])
    examination.append(info[1])
    if online_down(info):
        info = get_offline_info(row)
        global_info.append(info)
    else:
        global_info.append("")
webs["HISTORY"] = history
webs["EXAMINATION"] = examination
webs["GLOBAL"] = global_info
```

SAVE DATA

```
webs.to_excel("./01_CIE_emedicine_info.xlsx")
```

9.3 Anexo III: Archivo «_02_word_analysis.ipynb» (I)

Imports

```
import pandas as pd
import re #regex
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords
#from googletrans import Translator
from deep_translator import GoogleTranslator
```

CONFIGURATION

Get data – Virtual Patients

```
webs = pd.read_excel("data/01_CIE_emedicine_info.xlsx")
webs.fillna('')

risk_words = pd.read_excel("data/risk_texts.xlsx")
```

FUNCTIONS

```
def tokenize_this(text):
    return list(filter(None,re.split(
        "\s+",re.sub("[,;.:\*\--_<>/\]\[)(?!%$°ª°'"\"+=\d\s]+", " ",
        text.lower()))))

def tokenize_and_filter(text):
    if not pd.isna(text):
        return [w for w in tokenize_this(text)
            if w not in get_stop_words()]
    return []

count_total= 0
def tokenized_words_to_str(text):
    global count_total
    count_total = count_total + 1
    if count_total % 10000 == 0:
        print(count_total)
    return " ".join(map(str, tokenize_and_filter(text)))

def get_all_words_one_list(list_of_lists):
    list_of_words = []
    for text_list in list_of_lists:
        list_of_words.extend(text_list)
    return list_of_words

def get_stop_words():
    stop_words = stopwords.words('english')
    try:
        f = open("stop_words/words_to_add.txt", "r")
        stop_words.extend(f.read().split("\n"))
    finally:
        return stop_words
```

TOKENIZE DATA

Tokenize - Virtual Patients words

```
webs["T_HISTORY"] = webs["HISTORY"].apply(tokenize_and_filter)
webs["T_EXAMINATION"] = webs["EXAMINATION"].apply(tokenize_and_filter)
webs["T_GLOBAL"] = webs["GLOBAL"].apply(tokenize_and_filter)
webs["TOKENIZED"] = webs["T_HISTORY"] + webs["T_EXAMINATION"] +
webs["T_GLOBAL"]

webs["RISKWORDS"] = risk_words["RISKTEXTS"].apply(tokenize_and_filter)
```

GET STATISTICS

Data for Virtual Patients

```
all_words = get_all_words_one_list(webs["TOKENIZED"])
print(f"After tokenization and filtering, there are {len(all_words)}
    words in total.")
print(f"For filtering, it has been used {len(get_stop_words())}
    words.")
print(f"Total unique words: {len(set(all_words))}.")
#print(pd.DataFrame(all_words).value_counts())
```

STORE DATA

Store - Virtual Patients

```
webs.drop(columns=['WEB','HISTORY','EXAMINATION','GLOBAL','T_HISTORY',
    'T_EXAMINATION', 'T_GLOBAL'], inplace=True)
webs.to_excel("./data/02_CIE_tokenized_info.xlsx", index=False)
```

9.4 Anexo IV: Archivo «_02_word_analysis.ipynb» (II)

CONFIGURATION

Get data – 2 Database

```
db_112 =
pd.read_csv("data/112/EM_112_ICD_selected.csv")[['DIA_CODIGO','PAC_SEX
O','LAB_RIESGO_VITAL','CALC_AGE']]
db_112.rename({'DIA_CODIGO': 'CIE', 'CALC_AGE': 'AGE', 'PAC_SEXO':
'SEX', 'LAB_RIESGO_VITAL': 'RISK'}, axis=1, inplace = True)

db_112_texts =
pd.read_csv("data/112/EM_112_ICD_selected.csv")[['ARB_VARIABLES' ...
    'VAR_Vómitos alimenticios/biliosos','INC_OBSERVACIONES']]
db_112_texts = db_112_texts.fillna("")
db_112_texts = db_112_texts.astype(str)

db_112['TEXT_ES'] = db_112_texts.apply(''.join, axis=1)
```

In case it has already been processed:

```
db_112 = pd.read_excel("data/02_112_patients_db.xlsx")
db_112 = db_112.astype(str)
```

FUNCTIONS

```
def days_to_year(days):
    try:
        return int(float(days)/365)
    except:
        return -1

def es_to_en(texto):
    return GoogleTranslator(source='es', target='en').translate(texto)

def list_es_to_en(texts_es):
    texts_en = []
    for i in range(int(136160 / 20)):
        in_es = " \n ".join(texts_es[i*20:(i+1)*20])
        in_en = es_to_en(in_es).split("\n")
        texts_en.extend(in_en)
        print(f'{i}')
    return texts_en
```

TOKENIZE DATA

Tokenize - 112 Database

```
# 1) Cambiamos etiquetas de riesgo {2.0: 1, 1.0: 0}
db_112['RISK'] = db_112['RISK'].apply(lambda value : int(value - 1))

# 2) Actualizamos la edad a días
db_112['AGE'] = db_112['CALC_AGE'].apply(days_to_year)

# 3) Eliminamos la varibale Trauma previo__NO, puesto que la
#    tokenizará en 3 distintas
db_112['TEXT_ES'] = db_112['TEXT_ES'].apply(lambda text :
```

```
    text.replace('Trauma previo__NO', ''))

# 4) Tokenizamos
db_112['TEXT_ES'] = db_112['TEXT_ES'].apply(tokenized_words_to_str)

# 5) Traducimos al inglés
db_112['TEXT'] = list_es_to_en(db_112['TEXT_ES'].tolist())

# 6) Tokenizamos y filtramos stop words
db_112['TEXT'] = db_112['TEXT'].apply(tokenized_words_to_str)
```

GET STATISTICS

Data cases of 112

```
print(f"In the 112-Database there are {len(db_112)} cases.")
```

STORE DATA

Store - 112 Patients

```
db_112.to_excel("./data/02_112_patients_db.xlsx", index=False)
```

9.5 Anexo V: Archivo «_04_ml_classification.ipynb» (II)

Imports

```
import pandas as pd
import sklearn as sk
import numpy as np
import nltk

nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer # instead
of CountVectorizer and TfidfTransformer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.model_selection import GridSearchCV
from pickle
```

SET SEED TO MAKE EXPERIMENTS CONSISTENT

```
np.random.seed(500)
```

GET DATA AND PRE-PROCESSING

IF lemmatized data has been stored

```
virtual_patients = pd.read_excel(
    "data/04_virtual_patients_db_lemm.xlsx")
virtual_patients.fillna('', inplace=True)

db_112 = pd.read_excel(
    "data/04_112_patients_db_lemm.xlsx")
db_112.fillna('', inplace=True)

one_cie = pd.read_excel(
    "data/03_virtual_patients_cie_692,76.xlsx")
one_cie.fillna('', inplace=True)

space_cases = pd.read_excel("data/04_casos_específicos.xlsx")
space_cases.fillna('', inplace=True)
```

Create merged database

```
db_112["F_TEXT"] = db_112["T_L_TEXT"]
virtual_patients["F_TEXT"] = virtual_patients["L_TEXT"]

virtual_patients_and_db_112 = virtual_patients
virtual_patients_and_db_112 =
virtual_patients_and_db_112.append(db_112)
```

IF lemmatized data has NOT been stored

```
virtual_patients = pd.read_excel("data/03_virtual_patients_db.xlsx")
db_112 = pd.read_excel("data/02_112_patients_db.xlsx")

virtual_patients.fillna('', inplace=True)
db_112.fillna('', inplace=True)
```

Text to lower case + Tokenization + Remove Stop words

(PREVIOUSLY DONE)

FUNCTIONS

```
def get_pos_tag(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

count=0
def lemmatize(text):
    global count
    print(count)
    count = count + 1
    lemmatizer = WordNetLemmatizer()
    return ' '.join([lemmatizer.lemmatize(w, get_pos_tag(w))
        for w in nltk.word_tokenize(text)])

def get_results(y_test,y_pred):
    print(confusion_matrix(y_test,y_pred))
    print(accuracy_score(y_test, y_pred))
    print(classification_report(y_test,y_pred))
```

Lemmatize individually and merge Databases

```
virtual_patients['L_TEXT'] = virtual_patients['TEXT'].apply(lemmatize)
db_112['L_TEXT'] = db_112['TEXT'].apply(lemmatize)
```

Merge databases

```
virtual_patients_and_db_112 = virtual_patients
virtual_patients_and_db_112.append(db_112)
```

Store data

```
virtual_patients.to_excel(
    "./data/04_virtual_patients_db_lemm.xlsx", index=False)
    [['CIE','AGE','SEX','RISK','L_TEXT']]
```

```
db_112.to_excel(
    "./data/04_112_patients_db_lemm.xlsx", index=False)
     [['CIE','AGE','SEX','RISK','L_TEXT']]
```

EXPERIMENT __

DIVIDE DATA IN TRAIN AND TEST SETS

```
X_train, X_test, y_train, y_test = train_test_split(
    virtual_patients_and_db_112.F_TEXT,
    virtual_patients_and_db_112.RISK,
    test_size=0.25,
    random_state=76
)
```

MODEL AND TESTING - MULTINOMIAL (Naive Bayes)

Create pipeline: TF-IDF Matrix, select n elements and define model

```
pipeline = Pipeline([('tfidf', TfidfVectorizer(min_df=5, max_df=0.7)),
                     ('selec',  SelectKBest(chi2, k='all')),
                     ('model', MultinomialNB())])
```

Search for best parameters

```
parameters = {'tfidf__ngram_range': [(1, 1), (1, 2)],
              'model__alpha': (1e-2, 1e-3)}
param_pipeline = GridSearchCV(pipeline, parameters, n_jobs=-1)
```

Create and Test Model

```
model1 = param_pipeline.fit(X_train, y_train)
y_pred = model1.predict(X_test)

get_results(y_test,y_pred)
```

MODEL AND TESTING - LINEAR SVC

```
pipeline = Pipeline([('tfidf', TfidfVectorizer(min_df=5, max_df=0.7)),
                     ('selec',  SelectKBest(chi2, k='all')),
                     ('model', LinearSVC())])

model2 = pipeline.fit(X_train, y_train)
y_pred = model2.predict(X_test)

get_results(y_test,y_pred)
```

MODEL AND TESTING - RANDOM FOREST

```
pipeline = Pipeline([('tfidf', TfidfVectorizer(min_df=5, max_df=0.7)),
                     ('selec',  SelectKBest(chi2, k='all')),
                     ('model', RandomForestClassifier(
                         n_estimators=100))])

model3 = pipeline.fit(X_train, y_train)
y_pred = model3.predict(X_test)

get_results(y_test,y_pred)
```

MODEL AND TESTING - PERCEPTRON

```
pipeline = Pipeline([('tfidf', TfidfVectorizer(min_df=5, max_df=0.7)),
                     ('selec',  SelectKBest(chi2, k='all')),
                     ('model', MLPClassifier(solver='lbfgs'))])

model4 = pipeline.fit(X_train, y_train)
y_pred = model4.predict(X_test)

get_results(y_test,y_pred)
```

MODEL AND TESTING - ENSEMBLE

```
from sklearn.ensemble import VotingClassifier
ensemble = VotingClassifier(
    estimators=[('nb', model1), ('ls', model2), ('rf', model3),
        ('pe', model4)],
    voting='hard')

ensemble = ensemble.fit(X_train, y_train)
y_pred = ensemble.predict(X_test)

get_results(y_test,y_pred)
```

STORE MODELS

```
pickle.dump(model1, open('web/static/models/model1.pkl', 'wb'))
pickle.dump(model2, open('web/static/models/model2.pkl', 'wb'))
pickle.dump(model3, open('web/static/models/model3.pkl', 'wb'))
pickle.dump(model4, open('web/static/models/model4.pkl', 'wb'))
```

LOAD MODELS

```
model1 = pickle.load(open('web/static/models/model1.pkl', 'rb'))
model2 = pickle.load(open('web/static/models/model2.pkl', 'rb'))
model3 = pickle.load(open('web/static/models/model3.pkl', 'rb'))
model4 = pickle.load(open('web/static/models/model4.pkl', 'rb'))
```

9.6 Anexo VI: Archivo «index.html»

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>TFG - Garrido Sevilla</title>
    <link rel="stylesheet" href="{{url_for('static',
filename='css/main.css')}}">
  </head>
  <body>
    <a href="/" style="color: inherit; text-decoration: none;">
      <div id="header" class="header">
        <img src="{{url_for('static', filename=
               'images/logoVSH.png')}}" alt="Virtual Space Health"
               class="logo">
        <h1 class="headerText">Trabajo Final de Grado</h1>
        <h2>Ignacio Garrido Sevilla</h2>
      </div>
    </a>
    <div class="app">
      <div class="card">
        <h1>Predecir riesgo</h1>
        <form action = "{{url_for('predict')}}" method ="POST">
          <label for="observations">Texto Libre:</label>
          <textarea name="observations" rows="5" cols="20"
               placeholder="Introduzca las observaciones del paciente
               en inglés">
          </textarea>
          <input type="submit" value="Predecir">
        </form>
        {% if result != undefined %}
        <h2>Clasificación de los 5 modelos:</h2>
        <div class="results">{{result}}</div>
        {% endif %}
      </div>
      <div class="card">
        <h1>Generar Paciente virtual</h1>
        <form action = "{{url_for('generate')}}" method ="GET">
          <label for="cies">Elija la patología del caso a
               generar:</label>
          <select id="cies" name="cielist">
            <option value="930.9">Abrasión ocular (cuerpo extraño) -
               930.9</option>
            <option value="716.9">Artritis aguda - 716.9</option>
            <option value="M1">Aumento presión intracraneal -
               M1</option>
            <option value="993.1">Barotrauma (bloqueo sinusal) -
               993.1</option>
            <option value="703.9">Caída de uñas tras EVA -
               703.9</option>
            <option value="478.19">Congestión nasal - 478.19</option>
            <option value="361">Desprendimiento de retina -
               361</option>
            <option value="843.9">Distensión de cadera -
               843.9</option>
```

```
            <option value="993.3">Enfermedad por descompresión –
               993.3</option>
            <option value="841.9">Esguince / distensión del codo –
               841.9</option>
            <option value="M2">Exposición tóxica (amoniaco) –
               M2</option>
            <option value="733.93">Fractura por estrés de la
               extremidad inferior - 733.93</option>
            <option value="365.22">Glaucoma agudo - 365.22</option>
            <option value="373.9">Infección en el ojo - 373.9</option>
            <option value="959.12">Lesión abdominal - 959.12</option>
            <option value="993.2">Mal de altura/hipoxia –
               993.2</option>
            <option value="389.9">Pérdida de audición - 389.9</option>
            <option value="692.76">Quemadura solar - 692.76</option>
            <option value="958.9">Síndrome compartimental agudo –
               958.9</option>
            <option value="990">Síndrome de irradiación aguda (ARS) –
               990</option>
          </select>
          <div class="risk_option">
            <input type="radio" id="riesgo" name="risk_option"
               value="Riesgo Vital" required>
            <label for="riesgo">Riesgo Vital</label>
            <input type="radio" id="no_riesgo" name="risk_option"
               value="No Riesgo Vital" required>
            <label for="no_riesgo">No Riesgo Vital</label>
          </div>
          <input type="submit" value="Generar">
        </form>
        {% if patient != undefined%}
        <h2>{{patient[0]}}</h2>
        <div class="results">{{patient[1]}}</div>
        {% endif %}
      </div>
    </div>
  </body>
</html>
```

9.7 Anexo VII: Archivo «virtual_patients_app.py»

```
import pickle
import random
import pandas as pd
from flask import Flask, request, render_template

app = Flask(__name__)

multinomial = pickle.load(open('static/models/model1.pkl', 'rb'))
linearsvc = pickle.load(open('static/models/model2.pkl', 'rb'))
randomforest = pickle.load(open('static/models/model2.pkl', 'rb'))
perceptron = pickle.load(open('static/models/model4.pkl', 'rb'))

predictions = 0

def prediction(text, model, patient):
        this_prediction = model.predict(patient)[0]
        global predictions
        predictions += this_prediction
        return f'{text}
            { "PRESENTA RIESGO" if (this_prediction == 1)
            else "NO PRESENTA RIESGO" } \n'

@app.route('/')
def home():
   return render_template('index.html')

@app.route('/predict', methods =['POST'])
def predict():
        patient_to_classify = request.form.getlist('observations')

        result = prediction('Multinomial:   ',
                multinomial, patient_to_classify)
        result += prediction('Linear SVC:    ',
                 linearsvc, patient_to_classify)
        result += prediction('Random Forest: ',
                randomforest, patient_to_classify)
        result += prediction('Perceptron:    ',
                perceptron, patient_to_classify)
        global predictions
        result += f'Ensemble:       { "PRESENTA RIESGO"
                if (predictions > 2) else "NO PRESENTA RIESGO" } \n'
        predictions = 0
        return render_template('index.html',
                result = result)

@app.route('/generate', methods =['GET'])
def generate():
        cie = request.args.get("cielist")
        risk = 1 if request.args.get("risk_option") == 'Riesgo Vital'
                else 0

        return render_template('index.html',
                patient = [str(cie_data.loc[cie_data['CIE'] == cie]
                        ['CONDICIÓN MÉDICA'].iloc[0]),
```

```
                        str(create_virtual_patient(cie, risk))
                               .replace(', ', '\n')
                               .replace('{', '')
                               .replace('\'', '')
                               .replace('}', '')])

if __name__ == '__main__':
   app.run()
```