Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño de un sistema de diagnóstico basado en embeddings para medicina espacial

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Hernán Collado Ponce

Tutor: Juan Miguel García Gómez

Director Experimental: Pablo Ferri Borredà

Curso 2020-2021

Índice

Contenido

1. Resumen

A principios de 2021 tres naves espaciales, el Hope Orbiter, el Tianwen-1 y el Perseverance, llegaron a Marte [1] 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 resueltas.

En este proyecto, diseñaremos un sistema de diagnósticos compatibles para la medicina espacial basado en la clasificación por similitud de embeddings de redes neuronales profundas. [(Ferri, 2020)]

En este trabajo se expone el desarrollo de un sistema de ayuda a la recomendación médica, en este caso predicción de posibles diagnósticos para un paciente durante una emergencia médica en viajes de exploración espacial.

Los casos en los que se podrían desarrollar incidentes durante los meses que pueden transcurrir en el viaje hacia marte en las personas, no son en ningún modo nimios, y no solamente en el transcurso del viaje sino también, durante la estancia en el planeta.

Estos sistemas podrían proveer de gran ayuda a los tripulantes de la nave así como dar seguridad ante problemas médicos que puedan suceder, además no es un tema baladí el apoyo psicológico que este tipo de sistemas asistenciales puede aportar a la tripulación en el transcurso de su viaje y este, es un punto a resaltar, pues en gran medida, uno de los mayores desafíos que se encuentran en la actualidad para la puesta en marcha y ejecución de estos viajes espaciales es el componente psicológico de los tripulantes dentro de la nave y las relaciones que suceden entre si durante el viaje de más de 100 días en entornos relativamente confinados.

Una de las ventajas en la que nos basamos para desarrollar este trabajo es la cantidad de información recopilada en emergencias médicas que pueden ser útiles para construir estos sistemas de predicción de diagnósticos médicos.

1.1.1 Palabras Clave:

Inteligencia Artificial, aprendizaje automático, k-vecinos, transfer learning,

Resum

A principis de 2021 tres naus espacials, el 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 seva atmosfera i la seva 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 estat 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 resoltes.

En aquest projecte, dissenyarem un sistema de diagnòstics compatibles per a la medicina espacial basat en la classificació per similitud de embeddings de xarxes neuronals profundes. [(Ferri, 2020)]

En aquest treball s'exposa el desenvolupament d'un sistema d'ajuda a la recomanació mèdica, en aquest cas predicció de possibles diagnòstics per a un pacient durant una emergència mèdica en viatges d'exploració espacial.

Els casos en què es podrien desenvolupar incidents durant els mesos que poden transcórrer en el viatge cap a mart en les persones, no són en cap manera nimis, i no només en el transcurs d'el viatge sinó també, durant l'estada al planeta.

Aquests sistemes podrien proveir de gran ajuda als tripulants de la nau així com donar seguretat davant de problemes mèdics que puguin succeir, a més no és un tema intranscendent el suport psicològic que aquest tipus de sistemes assistencials pot aportar a la tripulació en el transcurs del seu viatge i aquest, és un punt a ressaltar, ja que en gran mesura, un dels majors reptes que es troben en l'actualitat per a la posada en marxa i execució d'aquests viatges espacials és el component psicològic dels tripulants dins de la nau i les relacions que succeeixen entre si durant el viatge de més de 100 dies en entorns relativament confinats.

Un dels avantatges en la qual ens basem per desenvolupar aquest treball és la quantitat d'informació recopilada en emergències mèdiques que poden ser útils per a construir aquests sistemes de predicció de diagnòstics mèdic.

Abstract

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

The adaptation of health and life support systems to the space conditions of long exploration missions pose a challenge for the technologies not yet resolved.

In this project, we will design a compatible diagnostic system for space medicine based on the similarity classification of deep neural network embeddings. [(Ferri, 2020)]

In this work, the development of a system to aid medical recommendation is exposed, in this case prediction of possible diagnoses for a patient during a medical emergency on space exploration trips.

The cases in which incidents could develop during the months that may elapse on the trip to Mars in people, are not in any way trivial, and not only during the trip but also during the stay on the planet.

These systems could provide great help to the crew of the ship as well as provide security against medical problems that may occur, in addition, the psychological support that this type of assistance systems can provide to the crew during their trip is not a trivial matter. and this is a point to be highlighted, because to a large extent, one of the greatest challenges currently encountered for the start-up and execution of these space trips is the psychological component of the crew members within the ship and the relationships that happen to each other during the trip of more than 100 days in relatively confined environments.

One of the advantages on which we rely to develop this work is the amount of information collected in medical emergencies that can be useful to build these prediction systems for medical diagnoses.

1. Introducción

1.2 Objetivos

El objetivo de este trabajo es el de **entrenar un sistema "Recomendador de emergencias médicas para vuelos espaciales"**, es decir un sistema de ayuda que ofrezca un listado ordenado de posibles diagnósticos compatibles con la situación de los pacientes en vuelos espaciales tomando como posibles entradas datos del paciente como su edad, sexo, síntomas, una descripción textual de los hechos etc.

En principio los posibles diagnósticos serán los 10 más probables y esta información será presentada mediante una lista de diagnósticos junto a su probabilidad, de forma sencilla para el usuario para que sea fácil de usar y de entender.

2. Antecedentes

2.1 Fundamentos del problema:

2.1.1 Viajes Espaciales:

Desde que la humanidad se adentró en los confines del cosmos, allá por la década de los sesenta del siglo pasado, las ambiciones de la humanidad, así como los desafíos que el espacio han supuesto para nosotros no han parado de crecer.

Desde el primer vuelo tripulado por un astronauta en 1961, hasta el viaje a la luna, el conocimiento que tenemos de la tecnología en el espacio, el lanzamiento de cohetes y la supervivencia en entornos ingrávidos como en la estación espacial internacional, son pasos hacia delante en el siguiente desafío que afronta la humanidad, el planeta rojo, Marte.

Gracias a nuevos avances en cohetería espacial, que desarrollan la NASA, la ESA y compañías punteras como "Space X" o "Blue Origin", la ambición de hacer viajes espaciales, no solo de vuelta a la luna si no hacia otros planetas ha ido creciendo en la población.

Como por ejemplo los cohetes reutilizables que algunas compañías anteriormente mencionadas han desarrollado y que es posible que, en breves, permitan realizar los primeros vuelos a marte con tripulantes a bordo. Como mención la nave “Starship” de “Space X” parece la más prometedora de esta nueva remesa de naves y cohetes modernos reutilizables, con más posibilidades de permitir estos vuelos gracias a su capacidad de carga y reducido coste gracias a ser reutilizable, que permite aprovechar prácticamente toda la nave mediante un sistema de aterrizaje autónomo y así evitar el desperdicio del cohete. Solamente es necesario recargarlo de combustible y realizar mantenimiento para tener la nave operativa de nuevo.

2.1.2 Estancia en el espacio:

Otros de los grandes problemas a los que se enfrenta el desarrollo de estos viajes interplanetarios es la larga estancia en el espacio, la cual, no está exenta de provocar problemas a los astronautas que realicen los viajes.

Debido a aspectos evolutivos los seres humanos están bien adaptados al entorno terrestre, por lo tanto, realizar viajes ingrávidos y con posibles dosis de alta radiación pueden provocar efectos negativos en el cuerpo humano a corto, medio y largo plazo.

Los efectos negativos de viajar al espacio pueden ser de atrofia muscular y alteración de funciones del aparato circulatorio, además se produce una redistribución de fluidos por efectos de la ingravidez que pueden desarrollar otros problemas como la alteración de la visión, así como el gusto etc.

Hay efectos de los vuelos espaciales que no son físicos, pero si psicológicos, y aunque estos últimos no se han analizado con gran rigor, si hay evidencia de efectos negativos en entornos similares al del espacio, pero en la tierra, como las estaciones de investigación en el Ártico, o submarinos.

Debido a grandes cantidades de estrés y adaptaciones del cuerpo humano a los cambios ambientales, pueden degenerar en ansiedad, insomnio y depresión.

Todos estos efectos tanto físicos como psicológicos, pueden ser resueltos o parcialmente solucionados con la investigación y el desarrollo de nuevas tecnologías, como evitar la radiación en el espacio, o los efectos de la ingravidez.

Este proyecto puede enmarcarse, aunque de forma modesta, dentro de las posibles soluciones para hacer los viajes espaciales más seguros y soportables para los astronautas. Tanto en el campo fisiológico como en el psicológico, ya que, mediante este sistema se da el soporte a la decisión durante circunstancias de emergencia médica en situaciones de aislamiento total de la tripulación por lo puede reducir el estrés en estos entornos hostiles y confinados. (Álvarez VM, 2020;5(08):1-11.)

2.2 Fundamentos Técnicos:

2.2.1 Aprendizaje automático:

El problema que nos ocupa puede ser resuelto de diversas formas, hay multitud de algoritmos que se pueden utilizar para realizar predicciones sobre una base de datos. Estos algoritmos predictivos pueden ser mejorados y aprender de forma iterativa, esto es el “machine learning”.

Se dice que un agente aprende cuando mejora su desempeño con la experiencia. Este tipo de programas lo que se busca es la mejora automática y continua sin la necesidad de escribir los programas explícitamente. Los programas resultantes de este aprendizaje deben ser capaces de generalizar comportamientos e inferencias para un conjunto más amplio de datos.

El aprendizaje automático puede ser visto como un intento de automatizar algunas partes del método científico mediante métodos matemáticos. Por lo tanto, es un proceso de inducción del conocimiento.

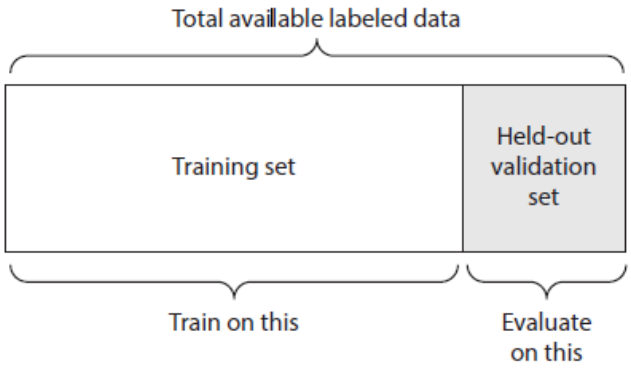
El aprendizaje automático tiene una amplia gama de aplicaciones, que incluyen motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, reconocimiento de imágenes, patrones y formas, juegos y robótica.[2]

2.2.2 Construcción de un sistema de aprendizaje automático:

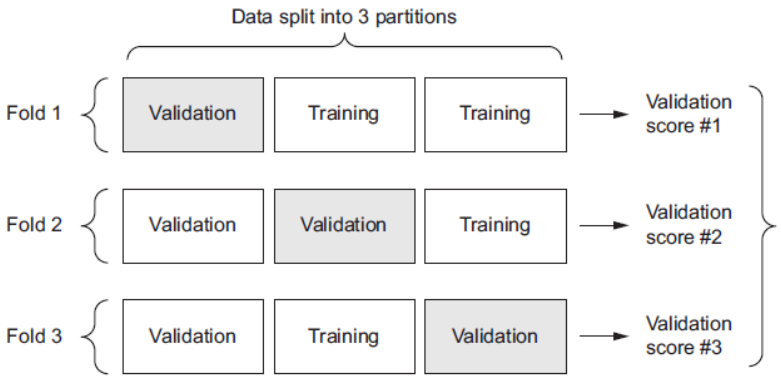
Antes de introducir ejemplos de sistemas de aprendizaje automático es recomendable conocer cuáles son los pasos que conviene seguir para construir un sistema de aprendizaje automático.

Para desarrollar un sistema de aprendizaje automático tenemos que pasar por una serie de técnicas relacionadas con la estadística y la programación, además de una serie de pasos que nos ayudaran a mejorar el propio algoritmo y su desempeño en la predicción de diagnósticos en nuestro caso en concreto.

1. **Definir el problema:** La tarea inicial y más crítica es la de encontrar cuales son las y salidas esperadas de nuestro sistema, para ello conviene hacerse una serie de cuestiones fundamentales a las cuales debemos dar respuesta, como:
    a. ¿Cuál es el objetivo? ¿Qué vamos a predecir?
    b. ¿Cuáles son las características objetivo?
    c. ¿Cuáles son los datos de entrada? ¿están disponibles?
    d. ¿Nos enfrentamos a un problema de clasificación binario o agrupamiento?
2. **Recopilar datos:** Este es el paso inminentemente posterior, al de la definición del problema. Es también un paso crítico con una influencia absoluta en el desempleo de nuestro modelo. Cuantos mejores datos mejor nuestro modelo y viceversa.
3. **Elegir una medida o indicador de éxito:** Para reconocer el desempeño de nuestro modelo es crucial definir los criterios de evaluación. Sin estos criterios bien definidos no podemos mejorarlo ni conocer si su desempeño está siendo exitoso.
4. **Establecer el protocolo de evaluación:** Cuando los objetivos están claros queda definir los criterios de evaluación para ello se pueden utilizar algunos de los más comunes, como:
    a. Set de validación "**hold out**": Consiste en mantener parte de los datos como un conjunto de datos de prueba. En el cual se entrenan el modelo con una fracción de los datos, ajustando sus parámetros con los datos de validación, y finalmente evaluando su eficacia usando como referencia los datos que hemos apartado:



b. Validación "**K-Fold**": En este caso se trata de dividir los datos en K particiones del mismo tamaño. En cada partición n, el modelo es entrenado con las restantes k-1 particiones y evaluado en la partición actual. La puntuación final es la media de las K puntuaciones obtenidas. Esta técnica es especialmente útil cuando el rendimiento del modelo es significativamente diferente del rendimiento del fragmento "entrenamiento prueba":



i.

5. **Tratar los datos:** Antes de comenzar a entrenar los modelos se debe hacer un tratamiento de dato de manera que puedan alimentar al modelo para ello se pueden seguir estas recomendaciones:

   a. **Datos perdidos:** Es frecuente en los datos que podemos obtener que parte de estos se hayan perdido. Debido a diversas causas ya sea por problemas en la propia recopilación de datos, espacios en blanco a la hora de realizar las encuestas, medidas no aplicables, desconocimiento de los datos a rellenar etc. Los valores vacíos aparecen con los indicadores "NaN" en nuestro caso ya que usaremos el lenguaje Python para desarrollar el modelo. Como estos modelos no pueden tratar los datos vacíos, en este caso podemos eliminar el dato si alguno de sus campos este vacío arriesgándonos a perder información relevante o de otra forma podemos rellenar estos datos utilizando alguna técnica de estimación ya sea por ejemplo la media de los demás si es un dato numérico y si es un dato categórico es decir no numérico como ejemplo si es alto, bajo, grande, pequeño, rojo, azul etc. Se puede entonces utilizar el valor mas frecuente para reemplazarlo por el faltante.

   b. **Escalado de Características:** Este paso es esencial en la fase de preprocesamiento, ya que la mayoría de los algoritmos de aprendizaje automático, tienen mucho mejor rendimiento cuando se los datos están en la misma escala. Para ello se puede **Normalizar o Estandarizar.**

   c. **División de datos en "Subsets":** Se dividen los datos en conjunto de entrenamiento, y conjunto de prueba. El objetivo es que el modelo pueda generalizar bien el con datos no procesados aun, poder usarse con conjuntos de datos no procesados, basados en sus parámetros internos ajustados mientras el modelo fue entrenado y validado.

i. **¿Como Aprende?**: Para conocer el proceso de "aprendizaje" de los sistemas de aprendizaje automático podemos utilizar el ejemplo de la regresión lineal. En este tenemos un numero de variables predictoras (explicativas) y una respuesta variable continua (resultado), entonces se intenta encontrar una relación entre ambas variables que nos permita predecir un resultado continuo. En este caso se puede utilizar la ecuación de la recta y usando métodos para estimar coeficientes podemos predecir el resultado utilizando datos nuevos. La fórmula obtenida contiene dos tipos de variables llamadas **pesos "W"** y **sesgos "B"**, estos valores serán los que se irán ajustando para reducir el error y mejorar las predicciones. Este proceso se produce a través de la comparación de la salida del modelo con los resultados correctos hasta obtener un buen modelo de predicción. Es un proceso iterativo donde cada vez vamos mejorando la salida.

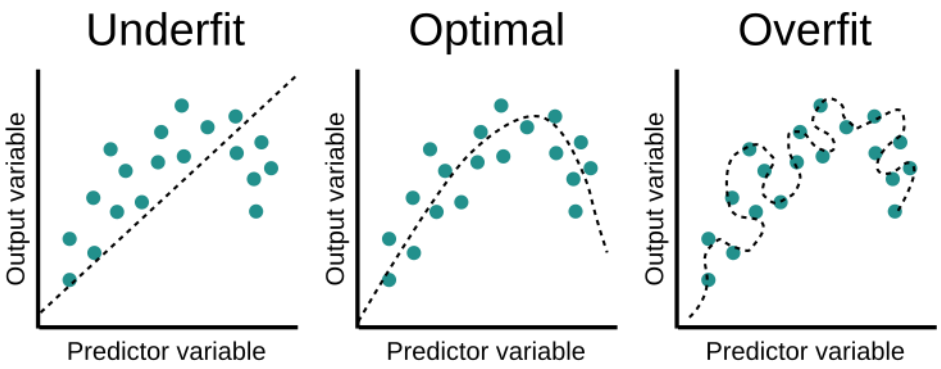
ii. **"Underfitting" y "Overfitting":** Uno de los problemas que podemos tener cuando estamos trabajando con el entrenamiento de modelos, es el conflicto entre generalización y optimización.

    1. La optimización es el proceso de ajuste de un modelo para obtener mejor rendimiento posible de los datos de entrenamiento.

    2. La generalización es la medida del comportamiento o tasa de acierto del modelo ante datos que no hayan sido procesados todavía. Nuestro objetivo será conseguir la mejor capacidad de generalización posible.

Al comienzo del entrenamiento, es normal que aparezca el "Underfitting" pues nuestro modelo todavía está desajustado, por lo tanto, nuestro modelo no es capaz de identificar patrones y producirá siempre pésimos resultados. Todavía queda aprendizaje por realizar debido a que aún no ha sido modelado con todos los parámetros importantes.

También podemos tener el resultado contrario, el "Overfitting", que también arroja malos resultados. Cuando ya tenemos suficientes datos de entrenamiento, la generalización reduce su progreso y las métricas de validación progresivamente comienzan a detenerse y luego se van degradando. Este resultado se puede deberse a que el modelo ha aprendido tan bien los datos de entrenamiento, que ya ha consolidado pautas que son demasiado específicas de los datos de entrenamiento que estamos utilizando actualmente pero que son irrelevantes y por lo tanto no aparecerán con nuevos datos.



[4]

Aunque estos resultados puedan ir apareciendo en nuestros modelos tenemos herramientas para reducirlos y obtener modelos correctamente ajustados. Algunas de las medidas son la **obtención de más datos** y/o la **regularización,** esta última, constriñe la cantidad de pautas que el modelo puede almacenar para que solo utilice las más relevantes.

6. **Desarrollo de un Modelo de Base:**

    a. El objetivo en este paso del proceso es, desarrollar un modelo de comparación que sirva de referencia, sobre el que mediremos el rendimiento de un algoritmo mejor y más ajustado. Para poder comparar correctamente nuestro modelo, los experimentos que realicemos deben ser comparables, medibles y reproducibles.

7. **Desarrollar un modelo mejor y Ajustar sus parámetros:**

    a. Primero debemos obtener un buen modelo, para ello podemos utilizar un método común, que es la validación cruzada. Para ello debemos establecer: Subdivisiones, que fragmentaran nuestros datos. Algún método para puntuar el desempeño del modelo y por último comprobar los algoritmos que queremos comprobar. Entonces comparando con los demás obtendremos el mejor algoritmo y será este el que ajustemos sus parámetros y optimizaremos para resolver nuestro problema.

    b. Este tipo de algoritmo de aprendizaje automático posee dos tipos de parámetros, unos son los que se generan en la fase de aprendizaje, y los otros son los “hiperparámetros”, aquellos que transferimos al modelo de aprendizaje automático. Una vez hemos seleccionado el tipo de algoritmo que vamos a utilizar, tenemos que ajustas los “hiperparámetros” para obtener los mejores resultados a la hora de realizar las predicciones. La forma más habitual de encontrar la mejor combinación de “hiperparámetros” se llama “Grid Search Cross Validation”. Este modelo devuelve un conjunto de “hiperparámetros” que se ajusta de la mejor forma al problema que estemos enfrentando. Una vez los tengamos determinados, ya tenemos el modelo listo para ser usado. Ya podemos hacer las predicciones pertinentes en el conjunto de datos de validación y guardaremos el modelo para uso posterior.

8. **Conclusión:** Con estos pasos ya podemos obtener un modelo de aprendizaje automático para realizar las predicciones sobre los datos que tengamos y así resolver el problema.

[5]

2.2.3 Usos del aprendizaje automático:

Un ejemplo en el que se usa el aprendizaje automático es, en el ámbito médico, como hemos mencionado anteriormente. Y se utiliza en diversos campos dentro de la medicina lo que facilita la tarea al personal sanitario y genera mejores resultados y una mejora sustancial para los pacientes, ya que el uso de estas tecnologías puede ser decisivo para su salud.

Algunas de las diversas áreas de implementación donde se hace uso del aprendizaje automático son:

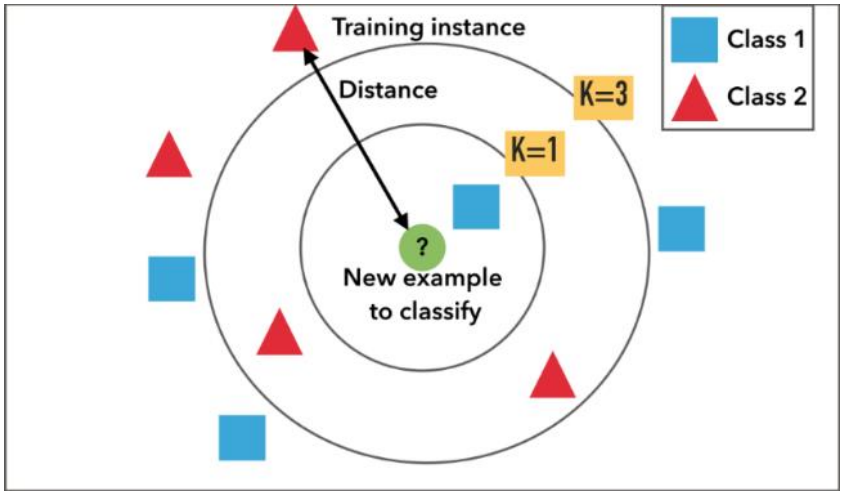
- **Pronóstico:** El aprendizaje automático tiene la capacidad de predecir con cierta seguridad la enfermedad a la que el paciente se enfrenta, si esta enfermedad puede ser crónica o la frecuencia con la que se puede producir, y además la esperanza de vida del sujeto. Además, se ha estudiado con gran profundidad en el campo de la oncología con el uso de estos algoritmos con resultados que remarcaban que el uso de estas técnicas de aprendizaje automático mejora entre un 15-25% las predicciones de pronóstico de cáncer. Además, se han utilizado en otras investigaciones las cuales predicen el riesgo de padecer ciertas enfermedades. Por ejemplo, usando un modelo de redes neuronales con millones de imágenes de ecocardiogramas de pacientes para identificar estructuras cardiacas, y estimar la función cardiaca y predecir factores de riesgo. [6]
- **Tratamiento:** Con el uso de modelos de aprendizaje automático, además, se puede identificar cual es el mejor tratamiento posible basándose en el historial de un paciente y sus características, además con estos métodos podemos descubrir el uso y el desarrollo de nuevos medicamentos.
- **Reducción de carga clínica y acceso al sistema de salud:** Todos los datos que hemos obtenido acerca de la información clínica nos ayudan en el desarrollo de estas tecnologías y algoritmos, ya que gracias a ellos se pueden entrenar y mejorar los sistemas de aprendizaje automático para que sean más fiables y eficientes. Gracias a esto podemos simplificar y agilizar todo lo referente a la atención médica y, además, reducir los costos de operación y el tiempo necesario para realizar tareas relacionadas. Algunos ejemplos de este uso es por ejemplo Wall et al, desarrolló un modelo capaz de evaluar y diagnosticar pacientes con autismo, este estudio visualiza aplicar el modelo con el fin de captar de manera preliminar los pacientes y posteriormente derivarlo al médico, gracias a este método se puede aliviar el trabajo clínico y de este modo hacer el sistema más eficiente. Estos algoritmos aún no están integrados en dispositivos electrónicos para el uso de los profesionales clínicos, sin embargo, novedosos estudios destacan la posibilidad de utilizar los algoritmos de aprendizaje automático y proporcionar una evaluación médica e implantarse en estos dispositivos electrónicos, de forma que estén al alcance de los médicos y personal sanitario. Aunque todas estas tecnologías no están disponibles para su uso in situ cabe recalcar sus posibles beneficios tanto en la mejora de acceso a los servicios sanitarios, la reducción de muchas consultas innecesarias, mejorar la comodidad de los pacientes, reducción de la carga laboral y la reducción de costes. (Roman, 2019)

Método de clasificación supervisada: K-Vecinos más cercanos (KNN):
[11]

Es un método de clasificación supervisada. Es un método de clasificación sin parámetros, que estima el valor de la función de densidad de probabilidad o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase C, a partir de información que se obtiene de un conjunto de prototipos. Y además en el proceso de aprendizaje no se hace ninguna suposición acerca de la distribución de las variables predictoras.

Es decir, el algoritmo clasifica cada nuevo dato proporcionado en el grupo que le corresponda, según tenga k vecinos más cerca de un grupo o de otro.

Esta clasificación la realiza a través del cálculo de la distancia entre el elemento nuevo a cada uno de los ya existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que ha de pertenecer el nuevo dato. Este grupo será el de mayor frecuencia con menores distancias.



1 Esquema de algoritmo KNN

¿Cómo funciona?

A diferencia de otros algoritmos de aprendizaje no supervisado, K-NN no genera un modelo a partir del aprendizaje con datos de entrenamiento, sino que el aprendizaje sucede en el mismo momento en el que se hace la prueba con los datos de test. Este método de aprendizaje se denomina "**lazy learning**".

Lazy Learning:

El lazy Learning o aprendizaje perezoso, es un método de aprendizaje automático en el que **la generalización de los datos** de entrenamiento, no se realiza hasta que se le consulta al sistema.

Es decir que solo obtiene la capacidad de obtener buenos resultados con datos nuevos hasta que se le realiza una consulta.

Esto se diferencia del “Eager Learning”, donde esta generalización sucede antes de recibir las consultas al sistema.

Uno de los principales motivos para utilizar el aprendizaje perezoso como en el algoritmo K-Vecinos más cercanos, es que el conjunto de datos se actualiza constantemente con nuevas entradas véase nuevos artículos en Amazon, nuevos videos en YouTube, nuevas series en Netflix etc.

Debido a la continua actualización de datos, los “datos de entrenamiento” quedarían obsoletos de forma relativamente rápida, sobre todo en áreas donde el conjunto de datos nuevos se incrementa velozmente como en la subida de clips de video a YouTube o plataformas similares de creación de contenidos.

Por tanto, el uso de estas tecnologías funciona de manera exitosa ante estos entornos cambiantes en los datos.

2.2.5 Redes Neuronales:

Las redes neuronales, son una familia de algoritmos de aprendizaje automático o “Machine learning”, se inspiran en el funcionamiento del cerebro humano.

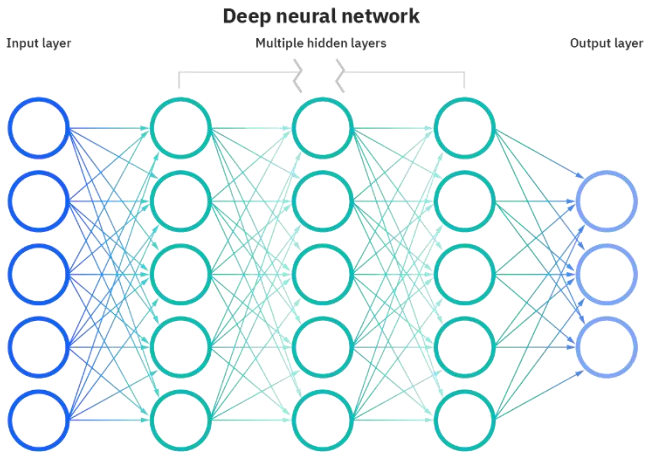
Están constituidas por un conjunto de nodos conocidos como neuronas, que están conectadas y transmiten señales entre sí, estas, se organizan en capas que van desde la entrada hasta generar una salida.

Las redes neuronales suelen tener tes componentes: la **capa de entrada**, donde tiene componentes que representan los campos de entrada de la red, tienen una o varias **capas ocultas**, con una unidad o unidades que representan el campo o los campos de destino. Estas unidades se conectan con fuerzas de conexión variables o

(**ponderaciones**). Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta las neuronas de la capa siguiente. Finalmente, la **capa de salida** donde se envían todos los resultados de las capas ocultas.

Estas redes aprenden examinando los registros individuales, generan una predicción para cada uno de los registros y realizan ajustes a las ponderaciones cuando se realizan predicciones incorrectas. Este proceso se va repitiendo hasta que se alcanzan los criterios para terminar, es decir cuando alcanza ciertas tasas de éxito.

Al comienzo todas las ponderaciones son aleatorias y por tanto las salidas o resultado de la red es pésimo. Entonces la red comienza a aprender iterativamente en el proceso de entrenamiento. Lo hace comparando las salidas que genera con resultados reales correctos. Esta información se pasa hacia atrás a través de toda la red, y en el proceso va cambiando las ponderaciones gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Y una vez entrenada con nuestros datos se puede generalizar para nuevos datos donde se desconoce el resultado correcto.[7][8]



[9]

2 Representación de una red neuronal con su capa/s de entrada oculta y capa de salida

2.2.6 Nuestro modelo:

Como hemos mencionado con anterioridad el objetivo de nuestro trabajo es realizar una aplicación que con unos valores de entrada específicos como puede ser la edad, el sexo, patologías previas de un paciente etc., seamos capaces de predecir el diagnóstico del mismo. Esta tarea está dividida en diferentes partes pues originalmente partimos de un trabajo otorgado por “Pablo Ferri”. En el cual, descrito de forma muy breve, se utiliza

una red neuronal profunda para predecir en casos de urgencia sanitaria, el nivel de amenaza para el usuario, la demora de la respuesta sanitaria y la jurisdicción (sistema de emergencia/ primaria ciudadano) en tiempo real.

Para hacer esto posible, es decir, utilizar una red neuronal pensada para generar un tipo de predicciones (amenaza para el usuario, jurisdicción y demora de la respuesta sanitaria), en otras que en nuestro caso serán **los posibles diagnósticos** que tenga el paciente, entonces nos basamos en el concepto del "**transfer learning**".

- **Transfer Learning:**

  Es una técnica dentro del Deep Learning que consiste en aprovechar el entrenamiento previamente realizado por una red neuronal y toda la información relacionada con la resolución de un problema para la resolución de otro diferente pero relacionado, es decir, que comparta algunas características con este.

  Un ejemplo práctico de este método es el realizado en la clasificación automática de imágenes. De forma general, este proceso se realizaría descubriendo una arquitectura de red neuronal que acelere el aprendizaje. A continuación, se otorgan valores de inicio a los parámetros que sirvan como variables de entrada para las capas de la red. Y en este momento es donde el transfer learning se muestra como un recurso economizador del tiempo y en gran medida del cálculo necesario. Con los datos obtenidos de una tarea similar, pero más genérica, evitas el utilizar costosas aproximaciones a un punto de partida óptimo. Posteriormente se van añadiendo capas hasta que los resultados obtenidos sean aceptables. Como hemos mencionado este método ofrece grandes beneficios en la creación de nuevas redes neuronales ya que reduce el coste y el tiempo de crear nuevas y además manteniendo la calidad de las predicciones.

Entonces y conociendo estos términos en los que nos encuadramos, nuestro propósito será a través de esta red neuronal original, utilizarla para generar predicciones de cuáles serán los posibles diagnósticos de nuestros pacientes.

3. Materiales

3.1 Datos:

Los datos que hemos utilizado son un conjunto de 1 244 624 EMCI [1] independientes del Departamento de Servicios de Salud de la Comunidad Valenciana, esta información son recopilaciones de datos que se adquieren durante una llamada telefónica y también después de esta. Se recogieron entre los años 2009 a 2012. Los datos utilizados fueron aprobados por el Departamento de Servicios de Salud de la Comunidad Valenciana para el uso en este proyecto, eliminando antes cualquier información que pueda revelar la identidad de las personas.

Como hemos mencionado los datos se obtuvieron en dos partes durante y después de la llamada:

Los datos durante la llamada son los datos demográficos, factores circunstanciales, características clínicas recopilados a lo largo de la navegación del árbol de clasificación y las observaciones del despachador de texto libre.

Los datos posteriores a la llamada incluyen diagnósticos médicos estandarizados por los códigos Internacionales de clasificación de enfermedades. Estos son los datos que vamos a utilizar para el entrenamiento y el test de nuestro proyecto.

Con estos datos se entrenó a la Red neuronal de la cual partimos nosotros para realizar nuestras predicciones, pues para nuestros datos en concreto, contamos con un archivo csv (comma sepparated values) para el entrenamiento.

Los csv como su nombre indica son archivos separados por coma de formato abierto y sencillo para representar datos en forma de tablas (similar a un Excel), en el cual las columnas se separan por comas o puntos y coma, y las filas se separan por un salto de línea.[10]

1 Incidentes de llamadas médicas

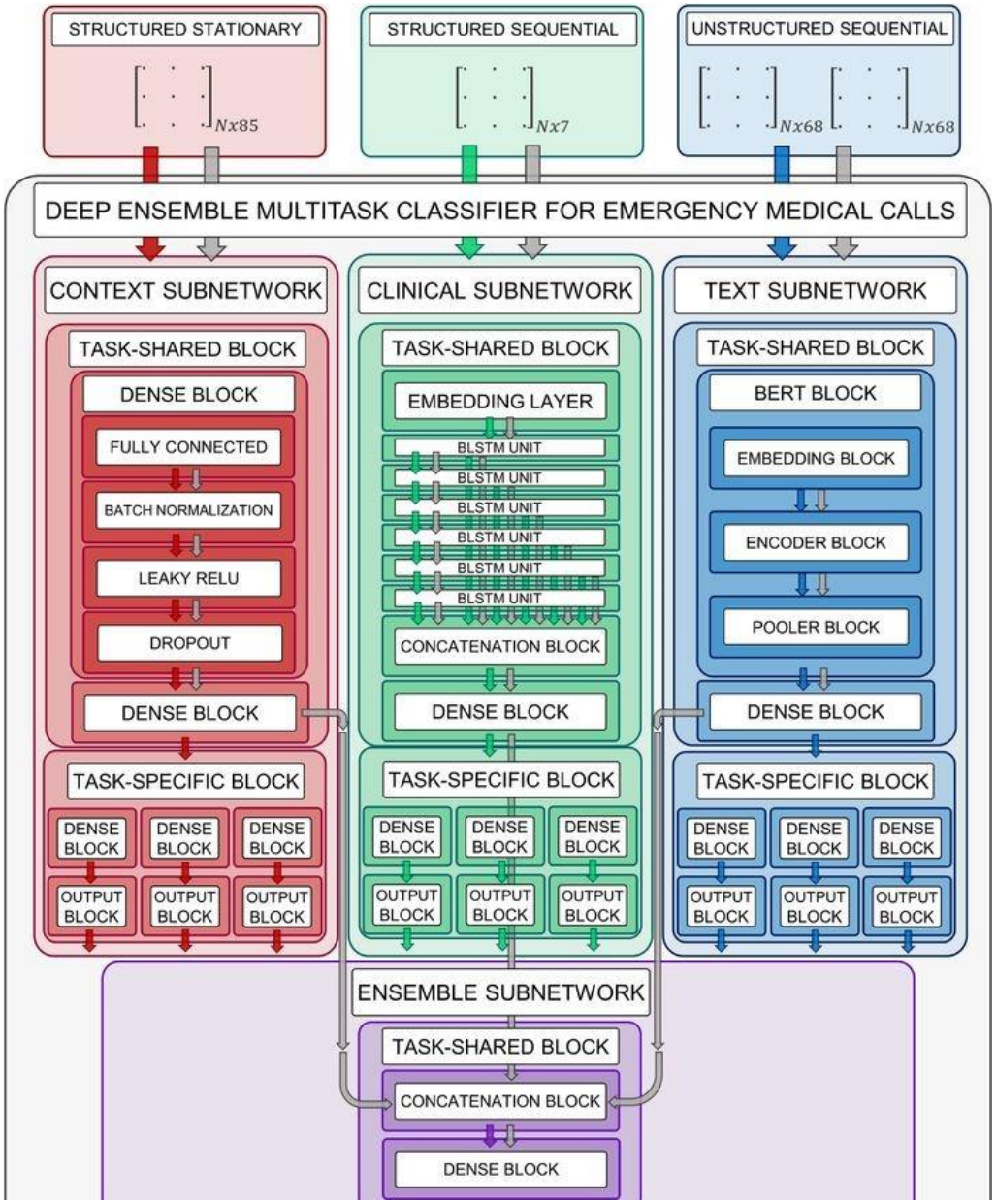
3 Imagen del aspecto de un dataset

Aquí tenemos un pequeño ejemplo de los datos vistos desde un editor (Excel).

Entonces nosotros tenemos un archivo csv de más de 130 mil filas. Cada columna es una característica que tiene el dato dentro de cada fila.

La columna o característica importante que vamos a utilizar para el entrenamiento es la de **GLOBAL_CIE_PRESENTE,** los cuales son así mismo una lista de uno o más posibles diagnósticos que tenemos asociados para cada caso. Y además utilizaremos un conjunto de datos resultantes de la predicción de la primera red neuronal.

Este conjunto de datos resultantes de la primera red neuronal, serán generados automáticamente utilizando los datos iniciales.



4 Representación gráfica de la red neuronal profunda que utilizamos inicialmente y extraemos sus predicciones en la última capa del "dense block".

En concreto vamos a utilizar un conjunto de 192 valores que genera la primera red neuronal como conjunto de características. Además de la columna de diagnóstico/s o **GLOBAL_CIE_PRESENTE (que será nuestra etiqueta),** que con esto harán las 193 columnas que utilizaremos para el entrenamiento y el test de nuestra IA.

Además de esta columna importante, tenemos otras tantas que cabe mencionar nos servirán para ejecutar la primera red neuronal. Esta está formada por 3 partes: la red de contexto, la red clínica y la de red de texto, serán explicadas detalladamente más adelante.

Algunas variables de datos que utiliza la **subred de contexto** están formadas por entradas estáticas y estructuradas, algunas de ellas son:

['NUM_IMP', 'BABY_MEMB', 'CHILD_MEMB', 'TEEN_MEMB', 'YOUNG_MEMB', 'ADULT_MEMB', 'ELDER_MEMB', 'NANAGE_RATIO', 'HOM_RATIO', 'MUJ_RATIO', 'NANSEX_RATIO', 'DIABETICO_RT', 'CARDIOPATA_RT', 'HEMOFILICO_RT', 'EPILEPTICO_RT', 'VIH_RT', 'ASMATICO_RT', 'ALERGICO_RT', 'PRB2_RT', 'PRBGR2_RT', 4 'PRBGR_RT', 'HIPERTENSO_RT', 'EPOC_RT', 'OBESIDAD_MORBIDA_RT', 'DEMENCIA_ALZHEIMER_RT', 'ALZHEIMER_RT', 'PARKINSON_RT', 'PSIQUIATRICO_RT', 'SIMULADOR_RT', 'HIPERFRECUENTADOR_RT', 'AVC_RT', 'DROGADICTO_RT', 'HABITUAL_RT', 'ALCOHOLICO_RT', 'NEOPLASICO_RT', 'TRANSPLANTE_RT', 'NANGR_RT', 'COM_112_SIN_COMUNICACION_OHE', 'COM_CENTRO_ATENCION_SIN_TECHO_OHE', 'COM_POLICIA_LOCAL_OHE', 'COM_POLICIA_NACIONAL_OHE', 'COM_GUARDIA_CIVIL_OHE', 'COM_POLICIA_AUTONOMICA_OHE', 'COM_SEGURIDAD_PRIVADA_OHE', 'COM_OTROS_CUERPOS_SEGURIDAD_OHE', 'COM_ENTORNO_PACIENTE_OHE', 'COM_ALERTANTE_ACCIDENTAL_OHE', 'COM_PACIENTE_OHE', 'COM_VECINOS_OHE', 'COM_OTROS_PARTICULARES_OHE', 5 'COM_ATENCION_PRIMARIA_OHE', 'COM_PAC_SAMU_DAUD_OHE', 'COM_HOSPITAL_PUBLICO_OHE', 'COM_HOSPITAL_PRIVADO_OHE', 'COM_AMBULANCIAS_OHE', 'COM_UHD_OHE', 'COM_OTRAS_INSTITUCIONES_SANITARIAS_OHE', 'COM_CRUZ_ROJA_OHE', 'COM_BOMBEROS_OHE', 'COM_PROTECCION_CIVIL_OHE', 'COM_OTRAS_INSTITUCIONES_EMERGENCIAS_OHE', 'COM_112_CON_COMUNICACION_OHE', 'COM_CODIGO_NO_RECONOCIDO_OHE', 'COM_NANALERT_OHE', 'INC_WEEKEND', 'INC_BKHOL', 'MONDAY_OHE', 'TUESDAY_OHE', 'WEDNESDAY_OHE', 'THURSDAY_OHE', 'FRIDAY_OHE', 'SATURDAY_OHE', 'SUNDAY_OHE', 'JANUARY_OHE', 'FEBRUARY_OHE', 'MARCH_OHE', 'APRIL_OHE', 'MAY_OHE', 'JUNE_OHE', 'JULY_OHE', 6 'AUGUST_OHE', 'SEPTEMBER_OHE', 'OCTOBER_OHE', 'NOVEMBER_OHE', 'DECEMBER_OHE']

Datos vacíos: Cuando tenemos el caso de los datos faltantes la red puede seguir funcionando excepto cuando faltan datos en las columnas:

- 'NUM_IMP': es decir, el número de personas involucradas.
- Todas aquellas variables que están relacionadas con el tiempo ya sea el día el mes, si es fin de semana o vacaciones.

Si tenemos datos **desconocidos**, esta red fallara lanzando un error. Por ejemplo, si agregamos un nuevo grupo de riesgo, el modelo dará un fallo al no reconocerlo.

La subred clínica

Las entradas que acepta la subred clínica son **estructuradas** y **secuenciales**.

En concreto son Nodos Clínicos en Árbol. Y tenemos muchos presentando aquí algunos de ellos:

Trauma previo__NO $_VOID_NODES_RECORD_$ Alteración de la conciencia__SI Trauma previo__SI Nivel de relación y contacto__PRESENTE Lugar del incidente__VIA PUBLICA Lugar del incidente__DOMICILIO Sin más información__SI Consecuencias de la clínica__PRESENTA MOVILIDAD Criterios código ICTUS__NO Deterioro nivel conciencia__SI Inconsciente__SI Disnea__SI Congestión nasal__NO Mareo__SI Fiebre__MAS DE 39 Cuadro Gastrointestinal (Náuseas,vómitos y/o diarrea)__SI Consecuencias de la clínica__IMPOTENCIA FUNCIONAL MIEMBRO Dolor__ABDOMEN Consecuencias de la clínica__AUSENCIA DE SANGRADO Inicio de la clínica__BRUSCO Número de heridos__1 A 3 Consecuencias de la clínica__PERDIDA SANGRE LEVE Dolor__TORAX Lugar del incidente__VIA URBANA Vómitos alimenticios/biliosos__SI Inicio de la clínica__PROGRESIVO Tipo de accidente__COLISION Tipo de accidente__ATROPELLO Inconsciente recuperado__SI Existencia de focalidad neurológica__SI …

El funcionamiento de la red cuando tenemos **representación faltante,** en este caso que el nodo se reporta, la red soporta este caso. Es el equivalente a **datos vacíos** en el caso anterior.

También acepta el caso en el cual el nodo es desconocido, es decir, el modelo puede tratar con nodos desconocidos sin dar errores.

Subred de Texto:

Las entradas que tiene son del tipo **desestructurado** y **secuencial.** Y para esta red es **texto libre**. Esta última red puede tratar correctamente **datos vacíos y datos desconocidos**, no dará ningún error cuando estos aparezcan como entrada.

Ejemplos:

“Hemorragia digestiva”,

“caída golpe de nariz pómulo hinchado”,

“varios días decaído flojedad pos mareos llama un $AOELLIDO_1951$”,

“hiperglucemia y aparente ictus”,

“dolor cabeza dolor general y mareada lleva días con tensión y azúcar descompensada”.

3.2 Herramientas:

Para este proyecto se han utilizado diversas herramientas para la escritura del código:

Entornos de Programación:

Se ha comenzado por el programa **Sublime Text**, un IDE de programación simple y sencillo de utilizar, que da soporte a multitud de lenguajes además de plugins y herramientas para facilitar la tarea de la programación, más adelante se ha añadido el uso del IDE **PyCharm**, este, es un editor mucho más potente, pues permite de forma nativa hacer “debugging” del código, con la ayuda que eso supone al análisis durante la ejecución y la inspección de variables además de su cambio de estado. Permitiendo así un trabajo más agradable y eficaz.

Lenguajes y librerías:

PYTHON

Enfocándonos en el lenguaje de programación que hemos utilizado, en este proyecto se ha utilizado el lenguaje de programación Python, en su versión 3, por su sencillez de uso, el amplio abanico de posibilidades que ofrece además de librerías para el desarrollo de redes neuronales y todos aquellos conceptos herramientas y características que estén relacionadas con el estudio y el desarrollo de la Inteligencia Artificial.

En cuanto a las librerías que hemos utilizado, se encuentran:

Pandas versión 1.2.3, una herramienta muy potente para el análisis de datos, en este caso se ha utilizado para leer los “datasets” (conjuntos de datos) y hacer modificaciones sobre estos.

Scikit-learn, es una librería que contiene multitud de funcionalidades para el uso del aprendizaje automático. Incluye varios algoritmos de clasificación, regresión y análisis de grupos. Entre ellos están “bosques aleatorios”, “K-means” y el que nosotros utilizamos **“K-neighbors”.** Con esta librería se pueden crear de forma sencilla este sistema de aprendizaje supervisado con una sencilla llamada y especificando los parámetros correspondientes.

Además, se utiliza para comprobar la eficacia del sistema con algunos métodos sean, **el F1 score o la confussion matrix o el accuracy score.**

RE (Regular Expressions), un módulo que provee de operaciones para encontrar coincidencias “matching”, en las expresiones regulares. Utilizado para encontrar patrones dentro de los datos y descartar aquellos que sean erróneos etc.

Control de versiones:

Para el control de versiones, un elemento fundamental a la hora de desarrollar software para evitar pérdidas de código, probar nuevas funcionalidades de forma segura y trabajar junto a compañeros o mantener el software en diferentes equipos, se ha usado **GIT.** Se ha elegido por su universalidad facilidad de uso y su flexibilidad para ser usado en diferentes proyectos.

Además, se ha utilizado **Tortoise Git,** un cliente grafico para Windows que se ha elegido por facilitar las tareas de control de versiones, pues elimina la necesidad de utilizar la consola de comando de Windows para usar Git, pues de esta forma es cómo funciona nativamente, y agrega una serie de interfaces gráficas, botones, ventanas y pestañas para realizar todas las operaciones de Git de forma más visual y sencilla.[12]

Hardware:

El hardware utilizado para el desarrollo de este trabajo y la ejecución del código ha sido el siguiente:

PC de sobremesa:

CPU: Intel I5 6500

GPU: Nvidia GTX 1060 6GB

RAM: 16GB

Almacenamiento: SSD 512GB, 1TB HDD

Portátil:

CPU: Intel I7 9750H

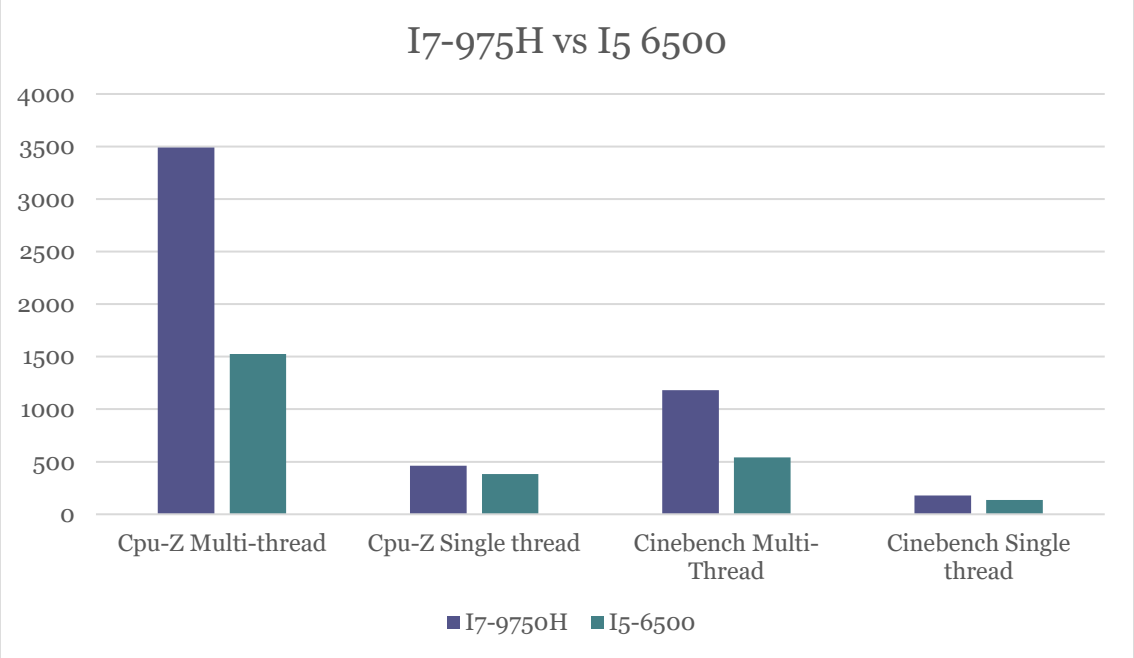
GPU: Nvidia GTX 1650 4GB

RAM: 8GB

Almacenamiento: SSD M2 NVME 512GB

Estos componentes son los causantes del rendimiento del sistema y si este se va a ejecutar de forma más o menos rápida. En concreto una de las piezas clave de el conjunto de componentes mencionados con anterioridad es el CPU o procesador.

En nuestro caso los equipos de que disponemos tienen ambos procesadores Intel, sin embargo, el equipo de sobremesa tiene menos potencia pues el procesador es de la generación 6 y además es de una gama inferior, ya que es un i5 con 4 núcleos y menor potencia que el portátil. Este último tiene un procesador de una gama mas alta ya que posee un procesador i7, además de ser este mas moderno, de la generación 9, y estar constituido por 6 núcleos su desempeño será mejor que el anterior.[13]



5 Resultados de Análisis de rendimiento de los diferentes procesadores Usados

En el anterior gráfico podemos observar cómo los resultados de diferentes pruebas de rendimiento, generan puntuaciones donde el i7 saca gran ventaja al i5, sobre todo cuando nos centramos en el rendimiento multinúcleo, es decir, cuando en el procesador se está haciendo uso de todos sus núcleos, la diferencia es más notable ya que el i7 tiene 6 núcleos en confrontación con los 4 del i5. Los resultados que utilizan un único núcleo no son tan dispares, aunque se puede observar como el i7 le saca ventaja al i5.

De estos datos se pueden extraer conclusiones como que el desempeño del i7 será mejor y por lo tanto los tiempos de ejecución del programa en concreto serán mejores.

Medidas a posteriori, han generado ya resultados favorables al i7 como por ejemplo el tiempo de ejecución a la hora de generar el dataset ultimo que se utiliza para el entrenamiento y el test de nuestro algoritmo. Siendo el del i5 de aproximadamente 45 minutos y el del i7 menos de 30 minutos.

4. Métodos

En esta sección se expondrán los métodos y técnicas utilizados en el análisis, la gestión, modelado etc., de los datos recibidos para el entrenamiento y el test de nuestro sistema de aprendizaje automático.

4.1 Clasificador utilizado:

En primer lugar y antes de mencionar que modificaciones hemos generado en los datos para el uso de nuestro sistema debemos mencionar que modelo hemos utilizado.

En nuestro caso hemos utilizado el modelo de K-Vecinos más cercanos o “K nearest neighbors” (KNN), el cual ya hemos explicado su funcionamiento anteriormente.

4.2 Análisis exploratorio y de calidad de los datos:

Para comenzar a utilizar sistemas de aprendizaje automático una de las fases más importantes es la de la calidad de los datos y si estos están formados correctamente. Y si en caso de que no lo estuviesen, que técnicas existen para mitigar estos efectos:

Eliminar datos perdidos:

Si en el análisis de datos se encuentran valores nulos o vacíos para ciertos campos del dato que aparecen en nuestro conjunto de datos como “NAN” (Not a number), es posible que la única opción que tengamos sea eliminar el mismo. Es decir, borrar toda la fila

donde se encuentra el dato faltante. Esta opción es la última a la que se debe recurrir ya que no se debería hacer si tenemos este caso de falta de algún campo en muchos de nuestros datos, ya que podemos estar modificando nuestro conjunto de datos. Para otras situaciones se debe recurrir a otro tipo de acciones.

Reemplazo de datos:

- **Por la media:**
    - Si tenemos un dato faltante en cuya columna haya valores no categóricos, es decir, valores numéricos una posible solución es remplazar el valor nulo por el calculo de la media de la variable completa, es decir, la media de todos los valores de dicha columna
    - En el caso que la columna cuyo dato faltante sea de tipo categórico, véase: grande, pequeño, rojo. amarillo etc. Podemos reemplazar el dato perdido con el valor categórico que mas se repite en dicha categoría.
    - Esta opción es preferible a la de la eliminación de los datos, pero agrega imprecisión a los datos ya que el dato nuevo que vamos a agregar es una conjetura, es decir, que puede ser tan cierto como falso.
- **Por un valor fuera del rango normal:**
    - Es decir, reemplazar el valor nulo por uno que sea mayor o menor al rango normal de aparición de esa columna. O por ejemplo si en con esa característica si es común resultados positivos reemplazarla con un valor negativo. Este método funciona bien con algoritmos basados en arboles de decisión.
- **Reemplazar por 0:**
    - Sustituir el valor nulo por un 0, esta técnica en concreto funciona correctamente con modelos de regresión y variables estandarizadas. Además, este método es válido incluso con variables cualitativas (no numéricas), si se guardan en el dataset de forma binaria.
- **Interpolar valores perdidos cuando formen parte de una serie de valores que hacen referencia al tiempo:**
    - Hay que mencionar que este método solamente funciona para valores cuantitativos. Por ejemplo, si el valor contiene visitantes diarios a un

lugar, se puede utilizar el promedio móvil de los últimos siete días o elegir el valor a la misma hora la semana anterior.

Mantener datos "vacíos":

A diferencia de en origen, los datos que dejamos en este tipo de modificaciones no se quedan como "NAN" en nuestro dataset, sino que se modifican para que se queden como datos vacíos o en nuestro caso listas vacías, pues este tipo de estructura de datos si las acepta la IA a la que tenemos acceso para realizar las predicciones, y el dato se queda con dos corchetes "[]".

Esta técnica la usamos mayormente a la hora de tratar los datos para que lo acepte la IA que estamos utilizando.

¿Cómo lo hemos hecho?

En primer lugar, debemos leer el conjunto de datos:

Esto lo hacemos utilizando la función "read_csv" dentro de la librería de Pandas.

Una vez tenemos leído el dataset comenzamos a realizar algunas transformaciones.

Como primer intento de generar buenos resultados con nuestro sistema de aprendizaje automático, vamos a utilizar solamente aquellos datos en cuya columna de diagnóstico solamente se encuentre un único diagnostico posible, más adelante se utilizarán todos los datos obtenidos del dataset, es decir con más de un único diagnostico posible.

El proceso de selección de datos para obtener únicamente aquellos con un único diagnostico posible será el siguiente:

1. Se añade una nueva columna en nuestro dataset cuyo valor contiene el número de diagnósticos de cada dato.
2. Entonces se seleccionan aquellos datos que solamente tengan un único diagnostico

3. A continuación, elaboramos una lista con los elementos únicos de los diagnósticos
4. A continuación, debemos transformar de texto a enteros cada diagnóstico, y guardamos la lista de diagnósticos originales con un numero asociada a cada diagnóstico, este número será el que remplazaremos en nuestro dataset.
5. Por último, se hace una pequeña limpieza de las filas que no tengan ningún diagnóstico es decir donde aparezca NAN en la columna. Y finalmente se retorna el dataset para poder comenzar a usar nuestro sistema de clasificación.

4.2 Formación del “DataFrame”:

Se ha mencionado como se genera el primer dataset para comenzar con las predicciones de los posibles diagnósticos y de como se han aislado en los datos recibidos aquellos que tienen un único diagnóstico, pero realmente el proceso de clasificación de los datos que realizamos nosotros utiliza un datset que se forma posteriormente al ya generado, entonces, ¿qué hacemos nosotros con el primer dataset y como generamos el dataset final?

Como hemos mencionado anteriormente, ya tenemos el dataset con datos que contienen un único diagnostico disponible, para generar el dataset ultimo que necesitamos primero debemos utilizar la red neuronal profunda proporcionada por Pablo Ferri.

¿Qué datos necesitamos?

Primeramente, hay que comentar que la Red Neuronal Profunda tiene tres partes o subredes bien diferenciadas las cuales vamos a dar uso:

1. **Subred de contexto:**
    a. Se ocupa de los factores demográficos y circunstanciales vinculados a un EMCI. Esta red tiene 5 variables de entrada que aparecen en nuestro dataset como listas de uno, ninguno o varios elementos que representan características de la persona o personas afectada/s.

b. **Los datos de entrada** que precisa son: el número de personas implicadas, la edad de esta o estas, su sexo, si pertenece a algún grupo de riesgo y por último quien ha alertado de la emergencia. Entonces los datos se obtienen de 5 columnas diferentes en nuestro DataFrame.

2. **Subred clínica:**

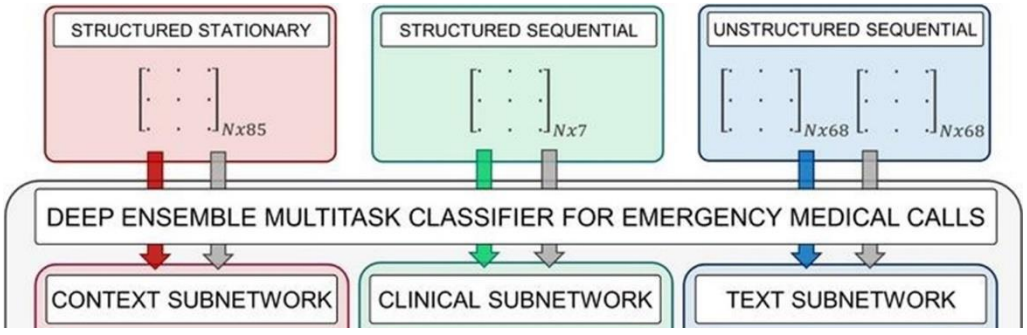
   a. Esta subred en particular trata de las características clínicas recopiladas durante la llamada. Es de importancia recalcar que las características notificadas en este punto se hacen de forma secuencial, pues su orden de registro es potencialmente informativo.

   b. **Los datos de entrada** pues que utiliza esta subred es realmente un único dato que contiene toda la información relevante durante la llamada. Todos estos datos se alojan en una única columna.

3. **Subred de texto:**

   a. La ultima de las 3 se ocupa de las observaciones del despachador de texto libre, es decir de un cuadro de texto libre donde se rellena información durante el EMCI.

   b. **Los datos de entrada** en este caso es un único dato que representa todo este texto mencionado con anterioridad y que se obtiene de una única columna.



6 Pequeña Ilustración de las 3 subredes

Una vez tengo los datos listos puedo hacer funcionar la red neuronal y comenzar a obtener resultados que nos serán útiles a la hora de formar el dataset final.

¿Cómo formamos el Dataset?

Para formar nuestro dataset lo que debemos hacer es primero obtener los datos generados por la red neuronal en una de sus capas específicas.

La capa especifica la cual vamos a hacer uso de sus datos generados es la capa denominadad “Dense Block” (véase figura 7), en la cual las salidas de cada una de estas subredes son vectores de 64 componentes diferentes que corresponden a valores intrínsecos dentro de la red neuronal.

Para formar nuestro dataset simplemente necesitamos obtener estos vectores del dense block y concatenarlos conjuntamente para formar un gran vector de 192 elementos al que le añadiremos finalmente el componente diagnóstico, para formar un vector de 193 elementos.

Entonces por cada entrada o fila en el dataset original obtendremos una fila formada por 193 columnas.

Aquí podemos ver un pequeño ejemplo de como se genera el dataset con la parte de la red clínica:

```
248     # --- Clinical inputs ---
249     try:
250       myInput['clinical'] = li.str_to_clinical_list(dataFrame["ARB_VARIABLES"][i])
251     except Exception as e:
252       print("Exception in clinical: ",e)
253       myInput['clinical']=[]
254     x1,x2,x3 = predictKNeighbors(**myInput, auxfiles = metadata)
255
256     x1 = x1.tolist()[0]
257     x2 = x2.tolist()[0]
258     x3 = x3.tolist()[0]
259     x1.extend(x2)
260     x1.extend(x3)
261     outcome.append(x1)
```

7 Generación del dataset usando la red neuronal inicial.

```
274   columns=[]
275   for i in range(len(outcome[0])):
276     columns.append(i)
277   finalOutcome = pd.DataFrame(outcome, columns=columns)
278
```

8 Generación de las columnas apropiadas para el dataset.

En la imagen 8, se añaden las entradas de la red clínica con:

myInput[“clinical”] = li.str_to_clinical_list(dataFrame["ARB_VARIABLES"][i])

Aquí se hace uso de la función str_to_clinical_list, esta hace un pequeño tratamiento de los datos para que acepte la entrada la red sin errores.

Mas adelante en la línea 254, se hace uso de la red neuronal con la llamada

predictKNeighbors, con esta llamada obtenemos una predicción de la red neuronal, la cual agregamos a nuestro dataset.

El resultado final es un DataSet de 129270 filas y 193 columnas, los títulos de las columnas no deben quedar vacíos pues los necesitamos para el correcto funcionamiento de nuestro sistema y la lectura y formación de el “DataFrame”, en este caso simplemente se han colocado de título números del 1 al 192 y finalmente una columna llamada “diagnosis” que hace los 193 componentes que necesitamos para comenzar a clasificar y testear nuestro sistema de clasificación.

	A	B	C	D	E	F	G	H	I	J	K	L
1	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57											
2	5,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,1											
3	80,181,182,183,184,185,186,187,188,189,190,191,diagnosis											
4	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
5	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
6	-0.00231021316722035 4,-0.00037602393422275 78,0.065762251615524 29,0.192052245140075 68,-0.00037102267378941 18,-0.00256745284423232 1,0.094350352883 3											
7	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
8	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
9	-0.00188681390136480 33,-0.00237859599929227 83,0.275913178920745 85,0.071694366633892 06,-0.00161821988876909 02,0.036539927124977 11,-0.00115693511907											
10	-0.00182404741644859 31,-0.00093171617481857 54,-0.00387535360641777 5,0.06492576003074646,-0.00107323739212006 33,-0.00366744236089289 2,0.235139787 1											
11	-9.074240369955078e-05,-0.00178413116373121 74,-0.00480366591364145 3,0.244837954640388 5,-0.00087032135343179 11,-0.00127395032905042 17,0.210560470 8											
12	-0.00196023657917976 4,-0.00213072891347110 27,0.264586687088012 7,0.085848033428192 14,-0.00181677751243114 47,0.015666097402572 632,-0.00043496966827											
13	-0.00170282064937055 11,-0.00022194265329744 667,-0.00362084060907363 9,0.255755424499511 7,-0.00052738859085366 13,-0.00278993695974349 98,0.125797957											
14	-0.00267180916853249 07,-0.00051351590082049 37,-0.00237349956296384 33,0.132456898689270 02,0.024601414799690 247,-0.00319309043698012 83,0.162981659 1											
15	-0.00149003835394978 52,-0.00179433682933449 75,0.193981468677520 75,0.137734785676002 5,-0.00129321846179664 13,-0.00046100586769171 06,-0.00074159546											
16	-0.00217937491834163 67,-0.00056449917610734 7,0.086263835430145 26,0.186290740966796 88,-0.00060598389245569 71,-0.00223772437311708 93,0.085316166281											
17	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
18	-0.00178621499799191 95,-0.00202606641687452 8,-0.00098879961296916,0.07525868713855743,-0.00079366192221641 54,-0.00092177995247766 38,-0.0001004533 4											
19	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
20	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
21	-0.00156346103176474 57,-0.00154646963346749 54,0.182654991745948 8,0.151888445019721 98,-0.00149177620187401 77,-0.00066974421497434 38,-1.96298951777											
22	-0.00188681390136480 33,-0.00237859599929227 83,0.275913178920745 85,0.071694366633892 06,-0.00161821988876909 02,0.036539927124977 11,-0.00115693511907											
23	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
24	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
25	-0.00257615046575665 47,-0.00114875833969563 25,0.168195545673370 36,0.120250329375267 03,-0.00093098520301282 41,-0.00141131947748363 02,0.043782189 48											
26	-0.00204853690229356 3,-0.00075297418516126 92,0.106765389442443 85,0.180592367935180 7,-0.00084094464546069 5,-0.00190796484078467,0.076281979680 06											
27	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
28	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
29	-0.00175415095873170 5,-0.00117704353760927 92,0.152893960475921 63,0.167565867304801 94,-0.00136960716918110 85,-0.00116610841359943 15,0.055955067 27											
30	-0.00211395579390227 8,-0.00065873668063431 98,0.096514612436294 56,0.183410003781318 66,-0.00072346429806202 65,-0.00207286048680543 9,0.080799087882 0											
31	-0.00188681390136480 33,-0.00237859599929227 83,0.275913178920745 85,0.071694366633892 06,-0.00161821988876909 02,0.036539927124977 11,-0.00115693511907											
32	-0.00221208459697663 8,-0.00051738024922087 79,0.081138432025909 42,0.187731117010116 58,-0.00054724351502954 96,-0.00232015666551888,0.087574720382 69											
33	-0.00230579543858766 56,0.042644217610359 19,-0.00244750222191214 56,0.207659184937078 74,-0.00052176322788000 11,-0.00305135059170424 94,0.148178994 65											
34	-0.00237563229165971 3,-0.00028178631328046 32,0.055511444807052 61,0.194932997226715 1,-0.00025354235549457 37,-0.00273231673054397 1,0.09886744618415											
35	-0.00277240783907473 1,-0.00086604547686874 87,0.137443184852600 1,0.128892570734024 05,-0.00057854369515553 12,-0.00190591171849519 01,0.057333469390 8											
36	-0.00260886014439165 6,-0.00110163947101682 42,0.163070112466812 13,0.121690705418586 73,-0.00087224482558667 66,-0.00149375153705477 71,0.046040743589											

9 Captura del dataset resultante, se aprecia el título de las columnas la cual hace referencia a los vectores y a la columna diagnosis

DataFrame:

Un objeto de tipo DataFrame se define como un conjunto de datos estructurados en forma de tabla donde la columna es un objeto de tipo Series, es decir todos los datos de una misma columna deben ser del mismo tipo. Los datos de las demás filas al contrario son registros que pueden contener distintos tipos de datos.

Los DataFrames contienen dos índices distintos, uno para las columnas y otro para las filas, además tienen la posibilidad de acceder a cada dato dependiendo del índice de la fila y la columna.

	First Name	Gender	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	97308.0	6.945	True	Marketing
1	Thomas	Male	61933.0	NaN	True	NaN
2	Jerry	Male	NaN	9.340	True	Finance
3	Dennis	n.a.	115163.0	10.125	False	Legal
4	NaN	Female	0.0	11.598	NaN	Finance
5	Angela	NaN	NaN	18.523	True	Engineering
6	Shawn	Male	111737.0	6.414	False	na
7	Rachel	Female	142032.0	12.599	False	Business Development
8	Linda	Female	57427.0	9.557	True	Client Services
9	Stephanie	Female	36844.0	5.574	True	Business Development
10	NaN	NaN	NaN	NaN	NaN	NaN

10 En el anterior ejemplo se puede observar los títulos de las columnas, y los índices de las filas.

Creación de un DataFrame a partir de un fichero CSV o Excel:

Para crear el DataFrame se puede usar la siguiente función:

Con el código read_csv(fichero.csv, sep=separador, header=n, index_col=m, na_values=no-validos, decimal=separador-decimal). Este devuelve un objeto del tipo DataFrame, con los datos del fichero **csv** usando como separador la cadena separadora. Los nombres de las columnas se añaden usando los valores de la columna **n**, definida en el header. Para los nombres de las filas se utilizan los valores de la columna **m**, si este no se especifica se añadirán número enteros incrementando en 1 y empezando por 0. En cuanto a los valores incluidos en la lista de **no-validos** se

sustituirán por NAN. Para los datos numéricos se utilizará como separado de decimales, el carácter especificado en el valor **separador-decimal**.

4.3 Gestión de los datos:

Una vez tenemos disponible el dataset último, para ejecutar nuestro sistema, debemos especificar como se va a entrenar el mismo.

Para entrenar un sistema de aprendizaje automático tenemos multitud de técnicas algunas mencionadas en el apartado de Fundamentos Técnicos, en el apartado de Aprendizaje Automático, se exponen algunas de estas técnicas comunes para el entrenamiento. En nuestro caso vamos a utilizar la técnica Hold-Out.

En resumen, este método utiliza una parte de los datos como entrenamiento y la otra parte como test o validación para mejorar sus resultados.

Como parámetros iniciales utilizaremos valores comúnmente utilizados en este tipo de métodos que serán un 70% para el entrenamiento y un 30% para el test.

4.4 Modelado:

En cuanto al modelado de datos hemos hecho algunas modificaciones.

Concretamente hemos **normalizado** los datos, esto es el proceso de ajustar los valores obtenidos en diferentes escalas respecto a una escala común.

A menudo previo a un proceso de realizar promedios.

Es decir, vamos a comprimir o extender los valores de nuestras variables para que estén en un rango definido.

Sin embargo, una mala aplicación de la normalización, o una elección poco concienzuda del método de normalización puede resultar en un desbaratamiento de nuestros datos y con este nuestro análisis.

Algunos ejemplos de normalización a menudo utilizados actualmente son:

- **Escalado de variables (MinMax Scaler):**

- En este caso, cada entrada se normaliza entre limites definidos:

$$Xnormalized = \frac{X - Xmin}{Xmax - Xmin}$$

- Este tipo de normalización tiene un problema, el cual comprime los datos de entrada entre unos límites empíricos (máximo y el mínimo de la variable). Esto quiere decir que si existe ruido en las muestras este va a ser ampliado
- Un ejemplo de problemas que puede acarrear este tipo de escalado es una distorsión de los datos, por ejemplo, si estamos midiendo la estabilidad de algún sistema véase, el de la calidad de conexión a internet y aplicamos el escalado una señal que a priori puede parecer estable se distorsiona sobremanera y daña nuestros datos. Por lo tanto, es recomendable evitar este método de normalización si tenemos delante señales estables.

- **Escalado Estándar (Standard Scaler)[1]:**
  - Otro método de escalado distinto puede ser el escalado estándar.
  - El proceso consiste en, a cada dato de forma individual se le resta la media de la variable y se le divide por la desviación típica.

$$Xnormalized = \frac{X - Xmean}{Xstddev}$$

  - Este método si que funciona con ejemplos como el anterior, es decir, con ejemplos que contienen señales estables.
  - Sin embargo, este método puede resultar erróneo, si se utiliza con datos que presenten variaciones anómalas (valores muy altos o bajos). Ya que los estadísticos que se usan (media y desviación típica), son muy sensibles a este tipo de datos anómalos.

En nuestro caso concreto vamos a utilizar el **escalado estándar**, el último de estos dos, y en concreto vamos a utilizar el escalador que viene integrado en la librería de **sklearn**. Este escalador utiliza una función para el cálculo ligeramente diferente de la mencionada anteriormente pues el denominador se sitúa la varianza y no la desviación típica, y aunque utilice estadísticos diferentes realmente vienen a medir los mismo, véase, es una medida de dispersión, definida como esperanza del cuadrado de la desviación de dicha variable respecto a su media. Es decir, a mayor valor de esta, mayor es la variabilidad de los datos. En cambio, a menor valor, aparecerá mas homogeneidad en los datos. [14][15]

La fórmula en concreto es:

$$Xnormalized = \frac{X - Xmean}{\sigma^2}$$

[16]

La vamos a utilizar con el comando integrado StandardScaler(), que escala utilizando esa formula aquello que le pasemos. En concreto nosotros usamos

StandardScaler.fit_transform(X_train)

Siendo X_train nuestra parte del dataset de entrenamiento. Y además con otra orden similar escalamos nuestros datos de test al igual que hemos hecho anteriormente.

4.4.1 Hiperparámetros:

Antes de exponer que valores hemos seleccionado para los hiperparámetros es necesario hacer algunas aclaraciones acerca de los hiperparámetros.

Para comenzar, un hiperparámetro, es un valor de configuración para nuestro modelo de aprendizaje (K-vecinos). Son valores que de forma general no se obtienen de los datos, por los que se suelen seleccionar de forma manual, en este caso nosotros seleccionaremos estos hiperparámetros.

La particularidad de estos valores es que no se pueden conocer a priori por lo tanto se deben hacer iteraciones en el sistema y tener métodos y valores que midan la eficacia de nuestro algoritmo para saber modificar y mejorar estos valores correctamente.

Para nuestro modelo de k-vecinos vamos a tener una serie de parámetros que seleccionar y especificar al modelo. Esto se hace en la creación de este pues dentro de los paréntesis se puede especificar los hiperparámetros:

En este pequeño recorte utilizamos el algoritmo en modo automático, el tamaño de la hoja en 30, la métrica es minkwoski aunque con el p=2 se está aplicando la distancia euclídea, y el número de vecinos es 301, los pesos los hemos puesto como distancia en este ejemplo. [17]

```
228
229     classifier = KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
230         metric_params=None, n_jobs=-1, n_neighbors=301, p=2, weights='distance')
231
```

11 Ilustración de cómo se crea el clasificador k-vecinos con sklearn.

Metric:

Indica el método por el cual se van a medir distancias en el clasificador, en concreto en esta llamada estamos usando la medida **Euclídea**, que como hemos mencionado con anterioridad, es la distancia entre dos puntos en línea recta.

Aunque se puede poner como distancia de minkowski, que es equivalente a la euclídea siempre que se cumpla que p = 2.

N_Neighbors:

Es el número de vecinos que vamos a utilizar en el clasificador. Este es un parámetro critico a la hora de generar los resultados.

La elección del número de vecinos **k** depende fundamentalmente de los datos. En general los valores grandes de **k** reducen el efecto de ruido de la clasificación. Pero crean límites entre clases parecidas.

El resultado y la eficacia de este algoritmo puede ser fuertemente degradada por la presencia de ruido en los datos o características irrelevantes, o si las escalas de las características no son consistentes con lo que uno considera importante.

En nuestro caso vamos a seleccionar el número de vecinos utilizando “cross validation”, “F1 score” y “Confusion Matrix”. Estos tres métodos los expondremos más adelante.

Weights:

Representa la función que se usa a la hora de hacer la predicción.

Los posibles valores que puede tener son:

- **“Uniform”:**
    - Los pesos de todos los vecinos están ponderados por igual.
- **“Distance:”**

    - Se puntúan los pesos por la inversa de su distancia. Es decir, que los vecinos más cercanos de un punto de consulta tendrán una mayor influencia que aquellos que estén más alejados.

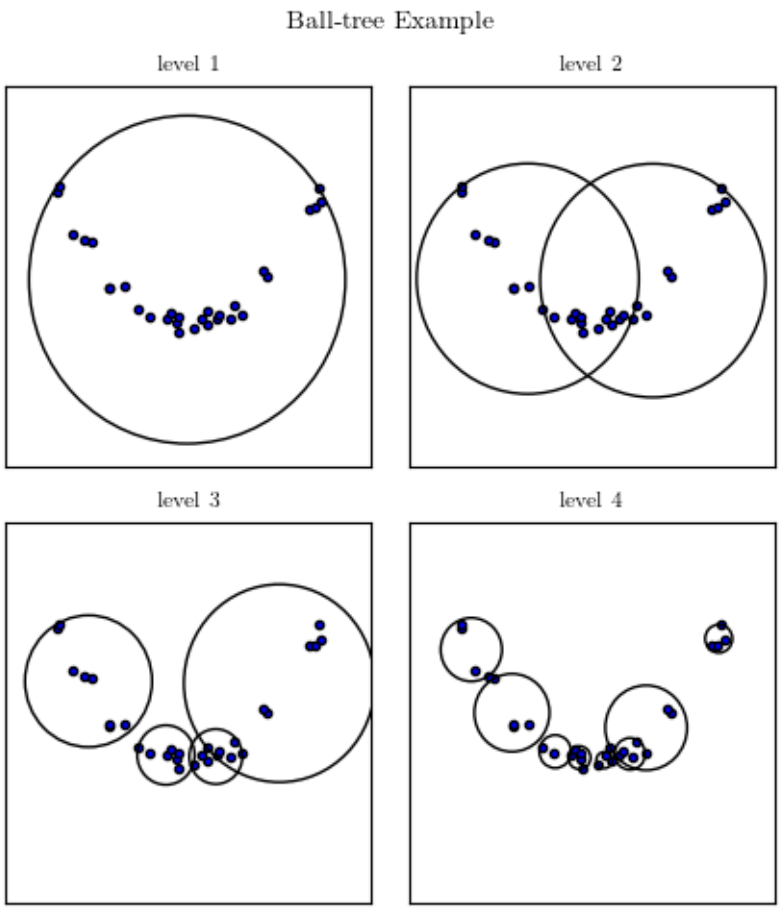
- "**Callable":**
    - Este último es diferente a los demás, pues es el propio usuario quien debe definir la función que acepte un array de distancias y que retorne otro array de la misma forma conteniendo los pesos.

Algorithm:

En este apartado se define el tipo de algoritmo que calculará los vecinos mas

cercanos.

- **'ball_tree'**
- **'kd_tree'**
- **'brute force' o "fuerza bruta"**
- **'auto':**
    - Este último tratara de definir el algoritmo más apropiado basándose en los valores aportados para ajustar el método.



12 Ilustración de distintas iteraciones de ejecución del algoritmo "Ball-tree"

Leaf Size:

El tamaño de las hojas que se especifica en este hiperparámetro será el que utilizará el algoritmo KDTree o BallTree. El tamaño de las hojas pude afectar a la velocidad de construcción y consulta, así como a la memoria que será necesaria para almacenar todo el árbol.

El valor óptimo de este dependerá en la naturaleza de los datos y del problema a resolver.

Number of Jobs (n_jobs):

Es el número de procesos que se ejecutan de forma paralela para la búsqueda de vecinos.

Valor P:

Este valor indica la potencia Minkowski. Es decir, Cuando p es igual a 1, equivale a "distancia Manhattan", si p vale 2, entonces tenemos la distancia Euclídea, y si p es igual a 3 o superior se utiliza la distancia de Minkowski.

Todas ellas explicadas anteriormente en el apartado de "El problema del clustering".

4.5 Evaluación del modelo:

Para que los resultados obtenidos por el modelo sean exitosos, es necesario tener buenos métodos de evaluación del comportamiento de este. Además, para poder perfeccionar el rendimiento del modelo si este resulta pobre.

Para ello en nuestro caso vamos a utilizar varias métricas que ya hemos mencionado con anterioridad siendo:

Precisión:

La precisión es una métrica la cual nos indica la cantidad de veces que el modelo acierta, o, cuantas veces el modelo arroja los resultados verdaderamente positivos y los verdaderamente negativos sobre el total de predicciones.

Recall:

Esta métrica, indica la cantidad, que es capaz de identificar el modelo. Es una métrica que compara el número de aciertos positivos, frente al número de total predicciones que positivas, las cuales pueden ser verdaderas positivas o falsas positivas.

F1 score:
Para este método de evaluación utilizamos el método incluido en el paquete de sklearn que nos proporciona una función que nos calcula este valor.

El cual se genera con la combinación de estos dos parámetros, el recall y el accuracy score.

5. Resultados:

En esta sección mostramos los resultados obtenidos.

5.1 Análisis exploratorios y de calidad de los datos:

En la creación de los datasets y en el momento de la ejecución de las redes neuronales encontramos muchos errores en los datos, tantos como datos faltantes o mal expresados, que la red neuronal no era capaz de procesar por tanto se han llevado a cabo acciones para evitar estas situaciones.

Aquí podemos observar algunos de ellos:

```
19 3,385010903,,486,-1,0,[None],1,['486'],0,[None],1,['486'],139398,"[['Trauma previo__NO'], ['AlteraciÃ³n de la
20 3,402010903,487,,,1,['487'],0,[None],0,[None],1,['487'],380500,"[['Trauma previo__NO'], ['Cuadro Gastrointes
21 3,404010903,487,,,1,['487'],0,[None],0,[None],1,['487'],651674,"[['Trauma previo__NO'], ['Fiebre__MAS DE 39
22 39,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,NO,,,1,18851.0,0.0,0.0,0.0,0.0,0.3353424657534248,0.06586301369863007,0.0,H,1.0,0.0
23 .0,108030140201.0,718147711.0,4247359588.0,0,1,THURSDAY,JANUARY,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,fieb
```

13 Pequeño ejemplo de datos vacíos.

En este se muestra uno de los valores que no acepta la red neuronal que se presentan como NaN y resulta en un error.

En los casos que resultaba en NaN la solución fue capturar el error utilizando un módulo Try, Catch y añadiendo una lista vacía para sustituirlo.

```
    for i in elems:
        try:
            final_list.append(int(float(re.sub(regex,"",i))))
        except Exception as e:
            #NaN exception
            final_list.append([])
    return final_list
```

14 Bloque Try Catch para recoger los errores de NaN

En este caso se observa como en los casos que da una excepción, es decir, llega un “NaN”, simplemente se añade a la lista final, una lista vacía “[]”. La cual si se acepta por la red neuronal.

Otro caso en el cual hemos tenido problemas al hacer uso de la red neuronal es con los datos del sexo los cuales no eran aceptados por la misma. Modificando las entradas haciendo una pequeña traducción hemos corregido el problema:

```
110    def translate_sex(sex):
111        if sex == "H":
112            return "HOMBRE"
113        return "MUJER"
```

15 Código de la traducción de sexo.

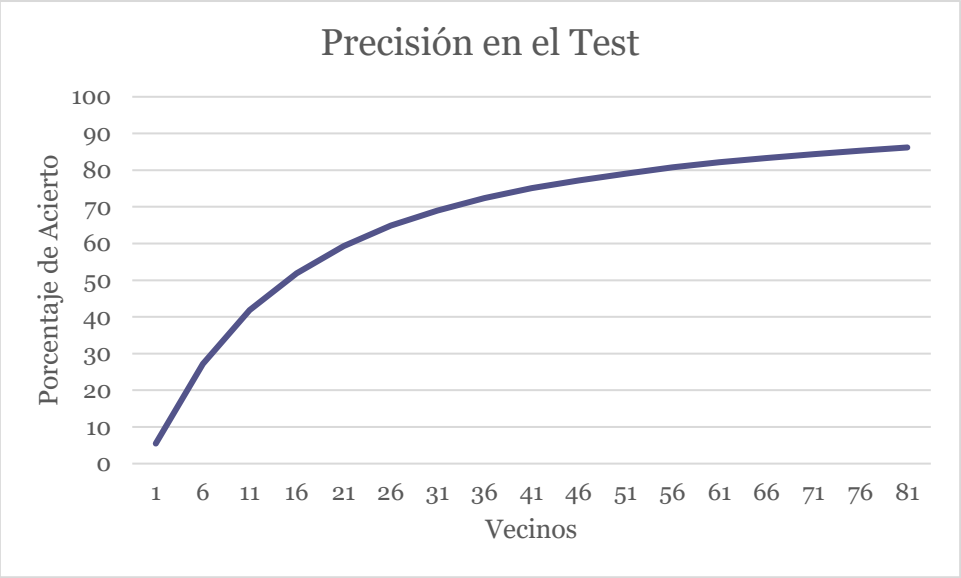
De forma breve hemos implementado un traductor que modificaba las entradas de “H” a “HOMBRE” y de “M” a “MUJER”. Debido a que en este caso las entradas iniciales “H” y “M” no eran reconocidas.

5.2 Evaluación del Recomendador:

En esta sección, vamos a evaluar el rendimiento de nuestro modelo utilizándolo como un sistema de recomendación.

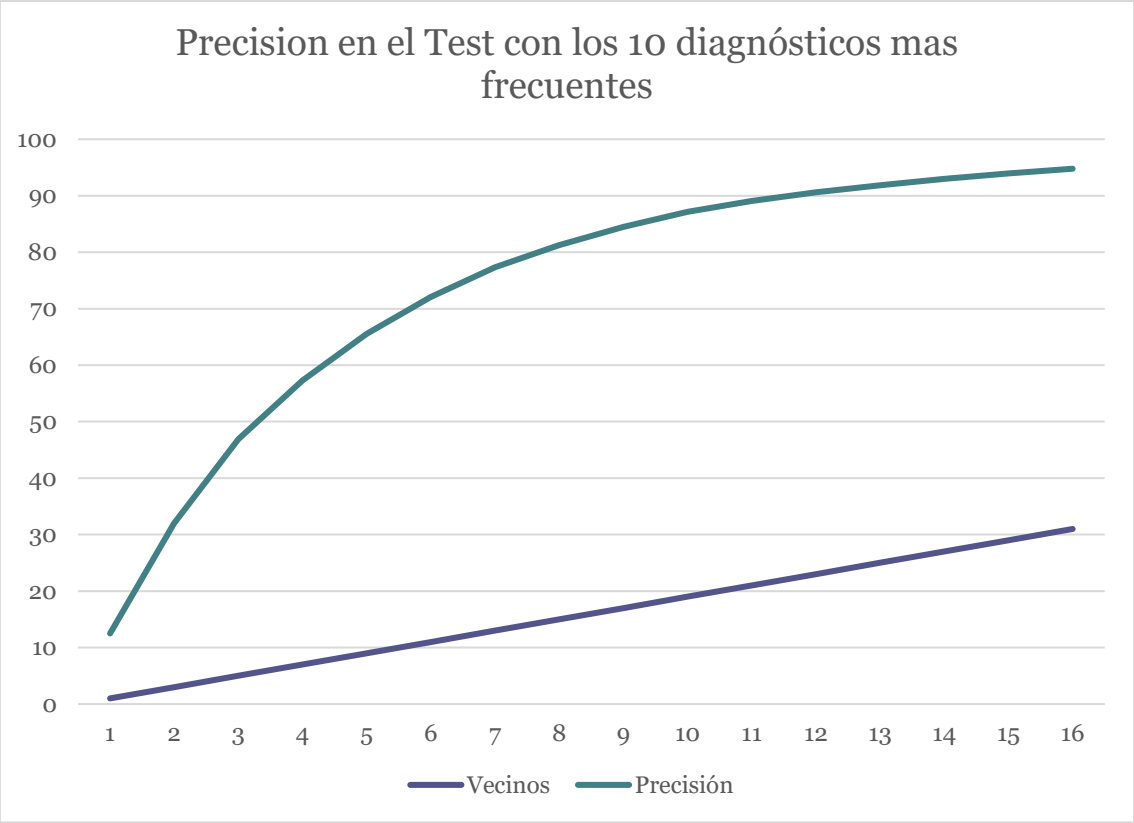
Este, funciona obteniendo los k posibles diagnósticos (que son nuestros vecinos en el gráfico), y, si en estos se encuentra el resultado correcto, se considera positivo. Los resultados obtenidos usando este método, han sido muy superiores y muestran una tendencia ascendente en relación con el número de vecinos.

Estos resultados se han obtenido en la evaluación del conjunto de datos que conforman el “test” denominado y_test en nuestro código.



16 Gráfico de la precisión del modelo respecto al número de vecinos utilizados.

Además, como el conjunto de diagnósticos es muy amplio y los resultados no son muy buenos (en torno al 70%-80%) hasta que llegamos a los 30 vecinos, o 30 diagnósticos, hemos realizado más pruebas con los datasets generados con los 10 diagnósticos mas frecuentes, facilitando la tarea a nuestro modelo debido a la reducción de la complejidad.



17 Grafico de la precisión del modelo con los datos de los 10 diagnósticos más frecuentes.

En este grafico se observa una gran mejora de rendimiento debido a que hemos utilizado los datos con los 10 diagnósticos más frecuentes.

Al reducir la dimensionalidad del problema el modelo es capaz de generar muy buenos resultados.

Con 10 vecinos, la precisión del modelo alcanza un valor de un 80% y si aumentamos hasta los 15 alcanzamos una precisión del 90%.

6. Discusión

6.1 Relevancia:

Las aportaciones más relevantes que podemos aportar con este trabajo, son las que se enmarcan en el campo de la Inteligencia Artificial, en concreto el campo del Transfer Learning.

Como hemos desarrollado en este trabajo, los sistemas de Transfer Learning obtienen muy buenos resultados en los campos del reconocimiento de imágenes, así como en el de lenguaje natural.

Sin embargo y como hemos constatado en este artículo, los resultados obtenidos en el proceso del transfer learning, o la transmisión del conocimiento, para las tareas de predicción de diagnósticos no son muy exitosas. Dado que el Transfer Learning tiene su gran aportación para el desarrollo de la IA en la facilidad que se tiene para utilizar modelos ya entrenados para resolver tareas relacionadas sin necesidad de comenzar un desarrollo profundo de un modelo y su entrenamiento. En este caso falla en esta tarea, pues no es capaz de transferir el conocimiento adquirido originalmente en tareas de en casos de urgencia sanitaria, predecir el nivel de amenaza para el usuario, la demora de la respuesta sanitaria y la jurisdicción, con el diagnostico de los casos utilizando los mismos datos de entrada.

Por el contrario, el sistema arroja mejores resultados a la hora de comportarse como un sistema de recomendación, pues es capaz de generar con alta tasa de acierto los 10 posibles diagnósticos. Alcanzando entre un 60% y un 80% de tasa de acierto en estos casos.

6.2 Limitaciones:

Las limitaciones que hemos encontrado en este desarrollo han sido tanto en la gestión y el tratamiento de los datos de entrada, para hacer posible el funcionamiento de la red neuronal “Pablo Ferri”, la comprensión de la propia red neuronal y su funcionamiento, así como en la dificultad de la predicción de los propios diagnósticos.

Esto, como ya hemos comentado, debido al overfitting presentado en los modelos de entrenamiento. Y la gran cantidad de diagnósticos a predecir.

Nos hemos encontrado con un problema de dimensionalidad, en la cual nuestro modelo no era capaz de predecir con eficacia el gran número de posibles diagnósticos.

6.3 Trabajo Futuro:

A consecuencia de los resultados obtenidos para las predicciones de nuestro sistema, algunas recomendaciones interesantes a realizar serian, utilizar otros modelos, como la regresión logística multinomial y además hacer una regularización incluyendo términos de penalización para intentar reducir el overfitting.

Así como, realizar un cribado de las características mas importantes a la hora de realizar la predicción, es decir hacer una reducción de la dimensionalidad eliminando características irrelevantes para mejorar las predicciones y mejorar los tiempos de ejecución del algoritmo.

Y aunque es cierto que utilizando nuestro sistema de recomendación de los 10 posibles diagnósticos si se han obtenido buenos resultados, quedan muchos retos por resolver y mejoras tanto en manejar con éxito la complejidad de los datos como en los modelos.

En el futuro y aplicando algunas de las recomendaciones expuestas en la sección de “Trabajo futuro”, este tipo de sistemas puede resultar en una gran ayuda para las misiones espaciales en el entorno de la asistencia médica y además como ayuda de los profesionales sanitarios a la hora de diagnosticar enfermedades, así como agregando nuevos sistemas de recomendación al tratamiento puede resultar en una gran herramienta para el futuro y la medicina moderna.

7. Conclusión:

En este trabajo se han hallado los principales problemas a resolver en la realización de un sistema Recomendador médico, para asistencia en vuelos espaciales.

De los datos obtenidos podemos remarcar la complejidad que ha supuesto su gran dimensionalidad, es decir, la gran cantidad de posibles variables de entrada, así como de las posibles soluciones, en nuestro caso, los diagnósticos a predecir.

Además, tras encontrar dificultades a la hora de predecir los resultados, y tras realizar un gran número de pruebas con diferentes modelos de predicción, se ha optado por un sistema de recomendación de diagnósticos.

El cual presenta un rendimiento aceptable y donde ha sido muy notable, en la tarea de recomendar los diagnósticos posibles, para los 10 más frecuentes.

Se puede concluir que el modelo garantiza un cierto éxito a la hora de recomendar posibles diagnósticos y que puede resultar una herramienta de gran utilidad en el transcurso de los viajes espaciales, así como para su uso en la medicina en hospitales y centros médicos en el futuro.

8. Bibliografía:

https://www.medrxiv.org/content/10.1101/2020.06.26.20123216v3

http://www.mars-one.com/

https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico

https://www.medigraphic.com/cgi-bin/new/resumen.cgi?IDARTICULO=95310#

https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48

https://rubialesalberto.medium.com/qu%C3%A9-es-underfitting-y-overfitting-c73d51ffd3f9

Álvarez VM, Quirós MLM, Cortés BMV. Inteligencia artificial y aprendizaje automático en medicina. Revista Médica Sinergia. 2020;5(08):1-11.

https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

https://www.viewnext.com/transfer-learning-y-redes-convolucionales/

https://es.wikipedia.org/wiki/Valores_separados_por_comas

https://pandas.pydata.org/

https://scikit-learn.org/stable/

https://es.wikipedia.org/wiki/Scikit-learn

https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos

https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos

https://www.iartificial.net/generalizacion-en-machine-learning/

https://docs.python.org/3/library/re.html

https://carmoreno.com.co/tutoriales/2016/04/14/TortoiseGit-Instalacion-y-uso/

https://datacarpentry.org/python-ecology-lesson-es/02-starting-with-data/

https://aprendeia.com/manipulando-datos-perdidos-en-python/

https://www.dummies.com/programming/big-data/data-science/machine-learning-choosing-right-replacement-strategy-missing-data/

https://aprendeconalf.es/docencia/python/manual/pandas/

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

http://rstudio-pubs-static.s3.amazonaws.com/429761_02ecd0fc2d4b4a54915318fc25fc7a38.html

https://empresas.blogthinkbig.com/precauciones-la-hora-de-normalizar/

https://latex.codecogs.com/legacy/eqneditor/editor.php?lang=es-es

https://economipedia.com/definiciones/varianza.html

https://definicion.de/varianza/

https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/

https://gadgetversus.com/processor/intel-core-i5-6500-vs-intel-core-i7-9750h/

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html