Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Creación de datasets y modelos predictivos basados en mensajería HL7

Trabajo Fin de Máster

Presentado por: Barrera Mayo, Joseba
Méndez Gutiérrez, Pablo
Vaquero Muñoz, Manuel

Director/a: De la Cruz Echeandía, Marina

Ciudad: Madrid
Fecha: 2022/09/21

Resumen

En los entornos médicos, se genera gran cantidad de información que no se suele emplear de manera predictiva sino descriptiva. En este trabajo se pretende hacer uso de esos datos para poder generar datasets a partir de mensajes HL7 y emplear técnicas de aprendizaje automático para ser capaces de predecir el alta por muerte de un paciente a partir del circuito completo de los episodios hospitalarios o de urgencias y el alta de un paciente (alta hospitalaria, alta voluntaria, muerte, etc.) a partir de los datos del conjunto de traslados hospitalarios de los pacientes. El enfoque se compone de los siguientes pasos: (i) Solicitud de los datos; (ii) anonimización; (iii) pre-procesado; (iv) creación de datasets; (v) la creación de varios modelos de aprendizaje automático y ajuste de sus hiperparámetros y del umbral de probabilidad para las predicciones, y (vi) la comparación de resultados entre los distintos modelos sobre los conjuntos creados para determinar la mejor solución. Los resultados muestran que una buena imputación de los valores ausentes y el ajuste del umbral de probabilidad son factores clave a la hora de mejorar el rendimiento de los clasificadores.

Palabras Clave: *Anonimización, aprendizaje automático, datos desbalanceados, mensajería HL7.*

Abstract

In medical environments, a large amount of information is generated that is not usually used in a predictive but in a descriptive way. In this work we intend to make use of these data to generate datasets from HL7 messages and use machine learning techniques to be able to predict the discharge due to death of a patient from the complete circuit of hospital or emergency episodes and the discharge of a patient (hospital discharge, voluntary discharge, death, etc.) from the data of the whole collection of hospital transfers of patients. The approach consists of the following steps: (i) data request; (ii) anonymization; (iii) pre-processing; (iv) creation of datasets; (v) creation of several machine learning models and adjustment of their hyperparameters and probability threshold for predictions; and (vi) comparison of results between the different models on the created subsets to determine the best solution. The results show that good missing value imputation and probability threshold adjustment are key factors in improving the performance of the classifiers.

Keywords: *Anonymization, HL7 messaging, unbalanced data, machine learning*

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

La gestión de los recursos hospitalarios es clave, sobre todo en momentos de alerta sanitaria como los que se han estado sufriendo debido a la pandemia. Se calcula que el Servicio Vasco de Salud (Osakidetza), que da servicio a toda la comunidad autónoma además de algunos pueblos colindantes, ha atendido más de 16M de consultas en el año 2019, según su informe anual.

Toda esta actividad médica genera una gran cantidad de datos, que permiten tener en todo momento al paciente controlado y tener un historial clínico sobre él. Esta información está altamente protegida y, para facilitar su operatividad, suele encontrarse estandarizada.

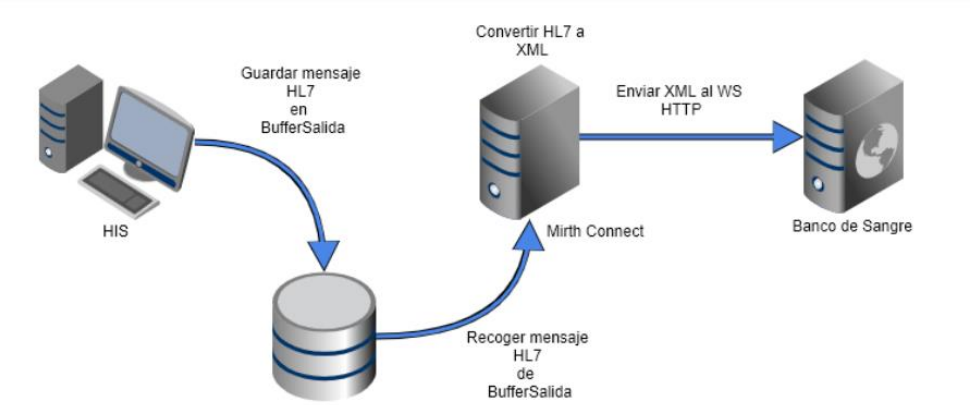
La mensajería HL7 es utilizada por miles de hospitales en todo el mundo como protocolo para el intercambio de información dentro del ámbito clínico. Este estándar permite comunicar datos entre equipos ubicados en un mismo hospital o con sistemas que se encuentren en diferentes organizaciones. Las normas HL7 actúan sobre una gran diversidad de datos clínicos como pueden ser el diagnóstico de un paciente, el ingreso en urgencias o el alta del mismo; pero también relacionados con la gestión del hospital como es el caso del cambio de cama de hospitalización o el transporte en ambulancia requerido por algunos pacientes.

El uso de esta tecnología permite comunicar aplicaciones de diferentes naturalezas; además, estandariza las comunicaciones de tal forma que si un equipo implementa este estándar es fácilmente integrable con otras aplicaciones y sistemas en el ámbito sanitario. Habitualmente se utilizan motores multiplataforma de integración como Mirth Connect[1] (ver Figura 1) que permite que los nodos puedan recibir, almacenar, procesar y propagar los datos a otras aplicaciones.

[1] Mirth Connect es un motor multiplataforma para HL7 que permite el envío bidireccional de mensajes HL7 entre sistemas y aplicaciones sobre múltiples formas disponibles, bajo la licencia Mozilla Licencia Pública (MPL) 1.1

Figura 1

Ejemplo de integración mediante Mirth Connect



Osakidetza dispone de una arquitectura propia de software a nivel centralizado, la cual se llama ***Event Manager***, que recibe información de las aplicaciones corporativas y propagan esos datos mediante mensajería HL7 a las aplicaciones y sistemas suscriptores, recibiendo en tiempo real esa información para poder realizar sus funciones.

Para suscribir una aplicación al ***Event Manager***, cada plataforma candidata debe publicar, al menos, un servicio web a través del cual recibe los datos de este de forma asíncrona, además de indicar mediante una XQuery los mensajes que necesita recibir. Por su parte el ***Event Manager*** recibe comunicados de las aplicaciones corporativas y revisa su lista de suscriptores y, dado que se trata de mensajería HL7 v3, filtra los suscriptores a los que los debe propagar a través de las XQuerys. Una vez obtenida la lista de suscriptores a los que debe reenviar, replica y propaga el mensaje invocando los servicios web de estos.

En este punto del proceso se pueden dar los siguientes casos:

- El mensaje se recibe correctamente por lo que el suscriptor, en base a sus necesidades, lo puede almacenar y procesar respondiendo ‘Ok’.
- El mensaje no se recibe porque o bien el servicio web del suscriptor está caído o hay algún error en el almacenado y procesado; por ello, el suscriptor no responde o responde ‘Ko’. En este caso, la información no se elimina de la cola del ***Event Manager***, y los mensajes que vienen posteriormente se van encolando detrás del inicial (el cual se intentará reenviar pasado un tiempo) con el objetivo de mantener el orden de procesado. Es muy importante mantener este orden al ser procesados en tiempo real pues, por ejemplo, si el objetivo de una aplicación es mantener un control del estado de las camas de un hospital, al procesar el alta antes que el ingreso la cama permanecería ocupada cuando debiera estar libre.

Además de para mantener monitorizado en todo momento al paciente durante su estancia en el hospital, la información suministrada por la mensajería HL7 puede ser utilizada, por ejemplo, para evaluar el funcionamiento de la clínica o para llevar a cabo predicciones o estudios.

En la actualidad, la inteligencia artificial está siendo utilizada en multitud de disciplinas entre ellas la medicina. El aumento de la disponibilidad de datos clínicos ha posibilitado que su uso en el ámbito de la salud haya experimentado un gran crecimiento durante los últimos años, más si cabe con la pandemia del coronavirus, que ha sido un gran foco de estudio durante el último año.

La aplicación de la IA en la medicina puede permitir una mejora en la calidad de los sistemas sanitarios y un trato más optimo a los pacientes como, por ejemplo, el uso de nuevos métodos de detección de enfermedades empleando clasificación de imágenes o la utilización de diferentes sistemas robóticos para ayudar en la logística de un hospital.

1.1 Motivación

Durante la pandemia derivada de la COVID19 se han vivido situaciones realmente complicadas en los sistemas de salud a nivel global. Aunque no todas, algunas organizaciones han estado al borde del colapso o han colapsado, no sólo en países subdesarrollados o en vías de desarrollo, sino también en países con sistemas de salud modernos y avanzados. Según el director general de la Organización Mundial de la Salud (OMS), Tedros Adhanom, esta situación ha provocado un colapso que deriva en la muerte de pacientes de otras enfermedades que, en circunstancias normales, podrían haberse prevenido, ya que cierto recursos clave se han destinado a sobrellevar esta situación. "*El rápido aumento de la demanda de instalaciones y profesionales sanitarios amenaza con dejar algunos sistemas de salud sobrecargados e incapaces de funcionar eficazmente*", destacaba Adhanom con sus declaraciones.

Por todo lo anterior, se puede aplicar la inteligencia artificial para prevenir posibles colapsos en los sistemas de salud, e igualmente mejorar la planificación de recursos de estos permitiendo generar diferentes predicciones con grandes volúmenes de información obtenidos de los históricos de mensajería HL7, que no deja de ser un registro de apoyo clínico y asistencial de lo que ha ocurrido durante el periodo registrado en un hospital.

Es decir, se podrían realizar diferentes predicciones en base a slots temporales, como por ejemplo la estancia en urgencias de un paciente en un hospital, el momento de su alta o el ingreso en hospitalización a su salida de urgencia, todo esto en base al histórico de

pacientes similares extraídos de este histórico de mensajería. Los resultados de estos modelos serían integrables en los sistemas de ***Business Intelligence*** del centro, con diferentes vistas predictivas que muestren los posibles escenarios a futuro del paciente. Lo mismo, es aplicable a otros recursos como quirófanos, unidades críticas, unidades de reanimación, o incluso boxes en la urgencia.

La motivación principal del proyecto es sentar las bases para implementar un sistema que ayude a mejorar la gestión de los centros de hospitalización, generando modelos basados en técnicas de Machine Learning que permitan realizar algunas predicciones sobre el uso de ciertos recursos críticos, investigando las posibles predicciones a partir de los datasets generados a partir del histórico de mensajería HL7. Algunas de estas predicciones podrían ser, el tiempo de hospitalización de pacientes, el riesgo de muerte, el destino del siguiente traslado (unidad de enfermería), etc.

En este caso se hace uso de los datos provenientes de la mensajería de Urgencia y Hospitalización del Hospital Universitario Cruces, ya que el miembro del equipo Joseba Barrera, es el fundador de la empresa Innotu Projects, y dispone de varios proyectos abiertos con la organización con diferentes suscriptores, que capturan la información (mensajería HL7), la almacenan y la procesan para resolver diferentes problemáticas. El Hospital almacena el histórico de mensajería de estos proyectos guardando al menos el último año para poder testear y desarrollar mejoras en caso de detectar nuevas necesidades. Es decir, si surgen nuevas funcionalidades se utiliza todo el histórico para generar nuevos indicadores que son validados con los propios datos.

1.2 Planteamiento del trabajo

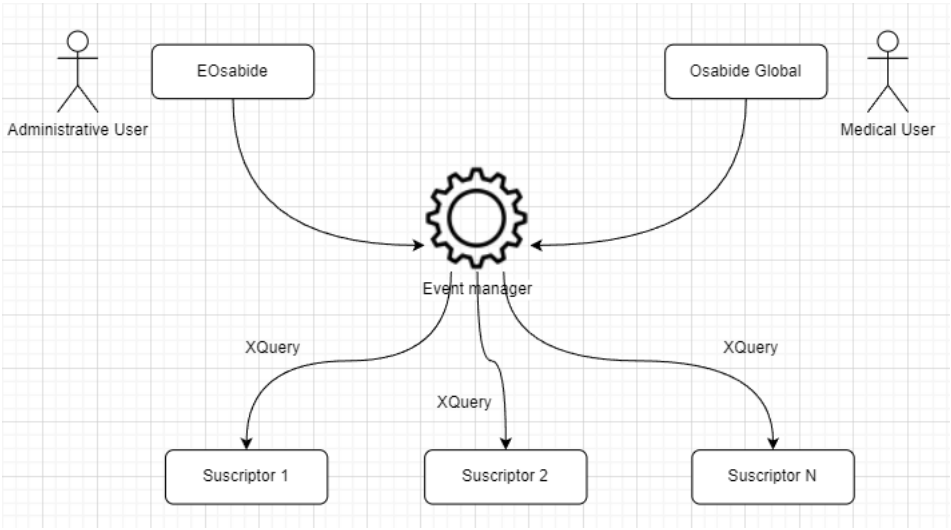
El servicio Vasco de Salud (Osakidetza) dispone de una arquitectura software llamada ***Event Manager*** (ver **Figura *2***) a nivel centralizado que recibe información de las aplicaciones corporativas de Osakidetza y propagan esta información mediante mensajería HL7 a los suscriptores.

Existen diversas aplicaciones que generan esta mensajería a través del ***Event Manager***, pero nos vamos a centrar en dos:

- EOsabide: HIS (Sistema de historia clínica) de la organización que permite registrar toda la información de los pacientes a nivel administrativo.
- Osabide Global: Estación clínica de la organización. Es decir, la aplicación mediante la cual se registra toda la información a nivel clínico.

Figura 2

Diagrama básico de suscripción al Event Manager



El ***Event Manager*** funciona como bróker de mensajes MQTT (protocolo de comunicación ***machine to machine***), es decir recibe una información, genera un evento y encola este mensaje en la cola de sus suscriptores. Una vez liberados los recursos para enviarlo, lo envía de manera asíncrona a través de servicios web SOAP[2].

Como se ha comentado, existen diferentes suscriptores que se suscriben a la mensajería que necesitan para poder realizar su función. Estos suscriptores publican servicios web que son dados de alta en el ***Event Manager*** y se encargan de recibir de forma asíncrona la información en tiempo real. Estos suscriptores tienen que responder a un estándar definido por Osakidetza y pueden estar implementados en diferentes tecnologías que Osakidetza considera como corporativas: .NET, Mirth, Java, etc.

Los suscriptores deben recibir sólo la mensajería HL7 estrictamente necesaria para cumplir con su función, indicando en su alta mediante una XQuery los mensajes que desea recibir:

- A nivel físico:
    - Toda una organización de servicios (Hospital y centros dependientes)
    - Todo un centro
    - Todo un edificio

[2] SOAP es un protocolo estándar que posibilita la comunicación entre las aplicaciones que se diseñaban con diferentes lenguajes y en distintas plataformas y aunque puede resolver diferentes tipos de formato como JSON, HTML, etc., en este caso propaga mensajes en formato XML.

    - Etc.

- A nivel asistencial:
    - Relacionada con la hospitalización
    - Relacionada con la urgencia
    - Relacionada con el bloque quirúrgico
    - Relacionada con laboratorios
    - Etc.

Dado que el Hospital dispone de ese histórico de mensajería y la experiencia de parte del equipo en el procesado de esta, creemos que es posible desarrollar un sistema de inteligencia artificial que nos sirva para crear modelos basados en Machine Learning que nos permitan realizar ciertas predicciones en el momento de recepción de nuevos mensajes HL7. Para ello es necesario realizar las siguientes tareas antes de ejecutar las predicciones:

- Estudiar los datos ofrecidos por Osakidetza en formato HL7:
    - Estudiar cada tipo de mensaje almacenado.
    - Determinar qué datos hay que eliminar y que datos hay que cifrar.
    - Determinar qué datos podemos extraer de cada tipo de mensaje para utilizarlos después en la creación de datasets, por ejemplo, timestamp de ingreso y alta, nos permitirá generar el tiempo del paciente en el área asistencial.
    - Determinar la estructura de esos datos en una base de datos relacional (ver Figura *3*).
    - Procesar los datos (2.5M mensajes HL7 aproximadamente) para rellenar la estructura anterior.
- Procesar los datos para generar el correspondiente dataset con datos normalizados en base a los modelos que se generarán. Aunque es necesario hacer un estudio previo, con el objetivo de poder monitorizar recursos clave es posible que el proyecto se centre en pacientes que tienen una secuencia de episodios compuesta por mensajes de urgencia, hospitalización y de forma opcional bloque quirúrgico.

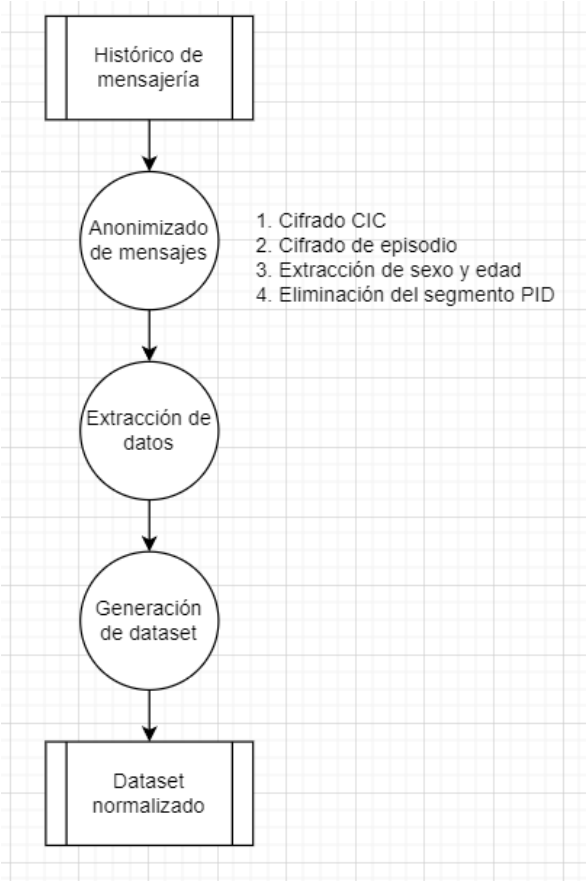
Para llevar cabo ese procesamiento es necesario desarrollar un software que (ver esquema explicativo en Figura 3):

- Anonimice[3] todo el histórico hasheando los datos CIC y episodio y eliminando el segmento que identifica al paciente (Segmento PID):
    - CIC, o código de identificación corporativo del paciente. Es el código que identifica de manera unívoca a un paciente dentro de toda la red de Osakidetza.
    - Episodio: Código que identifica de manera unívoca cada estancia de un paciente en un área asistencial del hospital. Existen diferentes tipos de episodio:
        i. Hospitalización
        ii. Urgencia
        iii. Consultas
        iv. Etc.
- Extraiga los datos relevantes que utilizaremos después para generar los datasets que alimentarán nuestros modelos. Es decir, debe generar una estructura con los datos que más tarde se utilizaran en los modelos realizados.
- Procese ese esquema de datos para generar datasets normalizados.
- Implemente un procesador que reciba un mensaje en tiempo real, extraiga los datos y los normalice en base al formato creado en los datasets. Esta línea de dataset será enviada a los diferentes modelos para generar las diferentes predicciones en base a al tipo de mensaje.

[3] Los datos se considerarán anonimizados en la medida que no exista una probabilidad razonable que cualquier persona pueda identificar a la persona física en el conjunto de datos.

Figura 3

Flujo propuesto



1.3 Estructura de la memoria

El presente documento de trabajo se encuentra estructurado en 7 capítulos, guiándose por la siguiente distribución:

- **Capítulo 1 – Introducción:** En este capítulo se explica de manera resumida el trabajo a realizar y el punto de partida del mismo, detallando la fuente de datos a utilizar y enumerando las diferentes tareas que hay que llevar a cabo antes de realizar las predicciones consideradas oportunas.
- **Capítulo 2 – Contexto y estado del arte:** Después de la introducción, se explican los diferentes algoritmos y modelos de aprendizaje automático e inteligencia artificial empleados para realizar las predicciones. También se presentan diferentes estudios y trabajos relacionados, mostrando los puntos de partida conceptuales que han sido tomados en cada uno de los diferentes bloques del trabajo.
- **Capítulo 3 – Objetivos y metodología de trabajo:** En este apartado se detallan las diferentes metas que se pretende obtener tras el desarrollo del trabajo. Igualmente,

se describen los procesos que se van a llevar a cabo para conseguir esos objetivos y como se va a evaluar la consecución de estos.

- **Capítulo 4 – Identificación de requisitos:** Tras explicar los objetivos establecidos, se especifica el trabajo previo realizado para poder efectuar el software, detallando la estructura de la información con la que se parte y explicando de manera resumida lo que se pretende realizar con los mensajes HL7 para poder generar los datasets.
- **Capítulo 5 – Descripción de la herramienta software desarrollada:** En este capítulo se describen los diferentes procesos efectuados para realizar el software; para ello se aportan capturas y diagramas que permitan un fácil seguimiento del programa.
- **Capítulo 6 – Evaluación:** En este apartado se muestran las diferentes métricas obtenidas para las predicciones realizadas, comentando y analizando cada una de ellas y estableciendo comparaciones entre los diferentes modelos empleados.
- **Capítulo 7 – Conclusiones y trabajo futuro:** Por último, se establecen una serie de conclusiones en base a los objetivos planteados y los resultados obtenidos, y se realiza un análisis de futuros trabajos que pueden surgir a raíz del presente proyecto.

En la parte final de la memoria, se incluye el detalle de las diferentes referencias enunciadas a lo largo del documento y el anexo con el código realizado durante el desarrollo. Además, se añade el trabajo en formato de publicación científica.

1.4 Organización del trabajo

El trabajo se ha realizado de manera grupal, concretamente por Joseba Barrera, Manuel Vaquero y Pablo Méndez; donde cada miembro, en función de su especialización previa, ha desarrollado algunos aspectos y colaborado en la consecución de otros.

Al tener un mayor conocimiento previo de los datos (mensajería HL7) y haber trabajado anteriormente con ellos, Joseba Barrera ha tenido un mayor peso en el TFM al comienzo de este, siendo clave en la obtención de los mensajes y en la extracción de valor de los mismos.

Por su parte, Manuel Vaquero ha concentrado una mayor cantidad de trabajo en la elaboración del programa necesario para el pre-procesado de los datos y la generación de datasets, ya que es la persona que más aptitudes tiene en la programación *.NET* y cuenta con una amplia experiencia en la materia.

Debido a su vida laboral y tener estudios previos en la materia, Pablo Méndez ha tenido un mayor peso en la realización del software necesario para ejecutar los modelos de predicción y las métricas que los evalúan, así como en la obtención de esos resultados.

Por otro lado, tanto la elaboración de la memoria como el análisis de resultados y conclusiones se ha realizado de manera conjunta trabajando todos con un reparto similar, debido a que son tareas que requieren un mayor nivel de atención, son claves para la consecución de objetivos y permiten la visualización de problemas en las otras áreas del trabajo.

En la Tabla 1, se detalla con porcentajes el trabajo realizado por cada uno de los integrantes en las diferentes fases que componen el trabajo.

Tabla 1

Tabla de distribución del trabajo

Tarea	Porcentajes		
	Joseba	Pablo	Manuel
Solicitud de datos	50%	25%	25%
Anonimización de datos	50%	25%	25%
Pre-procesado de datos	25%	25%	50%
Creación de datasets	25%	25%	50%
Creación de modelos predictivos	25%	50%	25%
Obtención de resultados	25%	50%	25%
Análisis de resultados y conclusiones	33,33%	33,33%	33,33%
Escritura de la memoria	33,33%	33,33%	33,33%

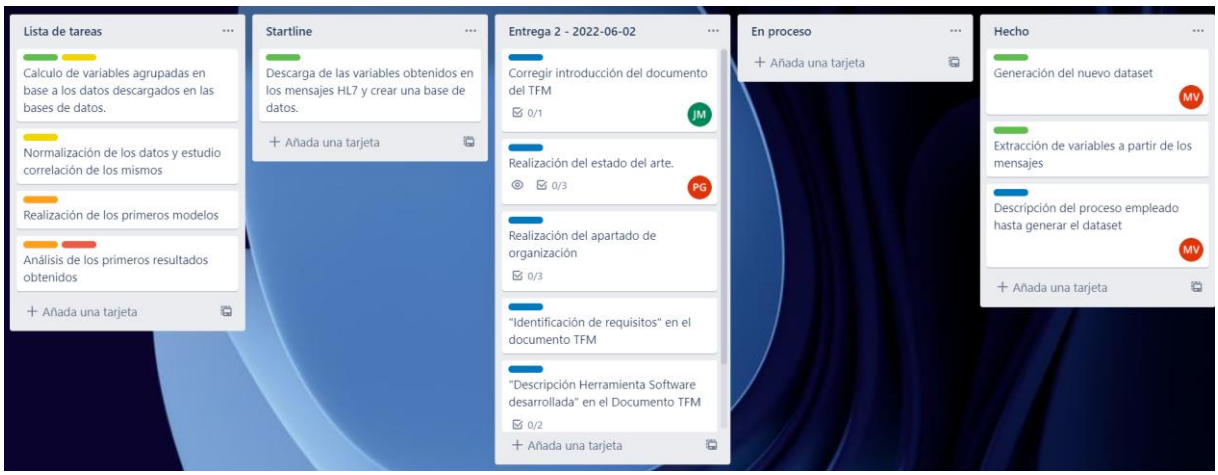
Para coordinarse y mejorar la eficacia del trabajo desarrollado, se ha empleado la metodología Scrum[4], que ha permitido una mejor cooperación y un conocimiento de las tareas a realizar por cada uno de los integrantes del grupo. Se ha elaborado una pizarra de trabajo (ver Figura 4), en la cual las tareas avanzan en las diferentes etapas que marcan el desarrollo de estas.

[4] Scrum es un marco de trabajo para desarrollo ágil de software que se ha expandido a otras industrias. Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de proyectos, caracterizado por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapar las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.

Figura 4

Pizarra de trabajo



Así mismo, se han mantenido reuniones semanales mediante la plataforma Microsoft Teams[5], donde se iba informando de los avances y los bloqueos encontrados durante el desarrollo del trabajo.

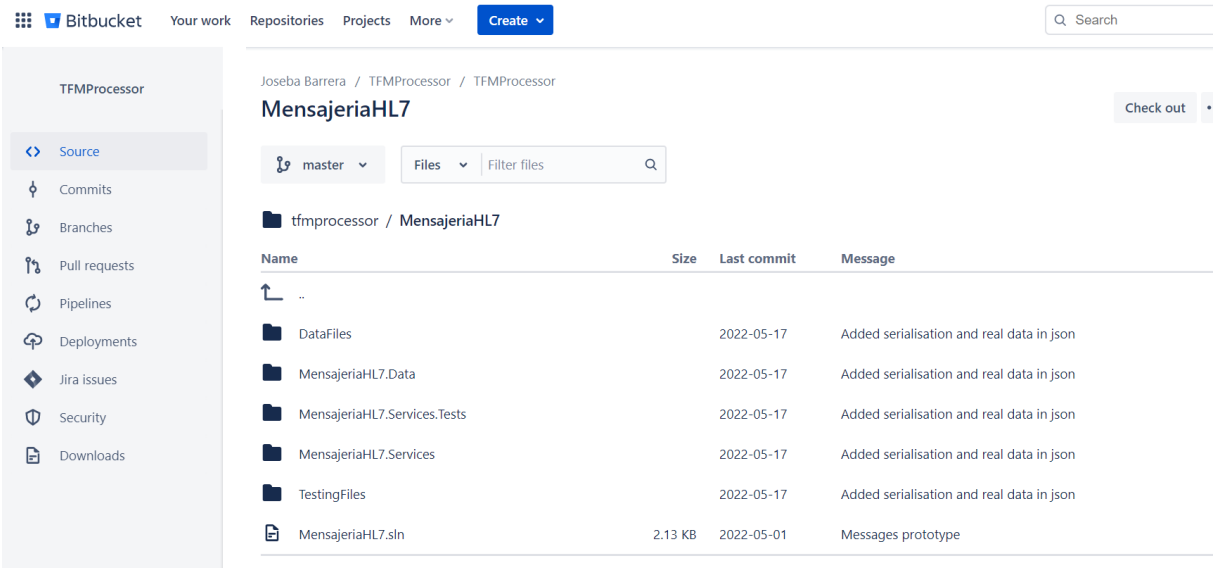
Por otro lado, para mantener la integridad del software y tener un control de los cambios realizados sobre el código, se ha trabajado sobre un repositorio Git[6], concretamente con Bitbucket (ver Figura 5)

[5] Microsoft Teams es una plataforma unificada de comunicación y colaboración que combina chat persistente en el lugar de trabajo, reuniones de video, almacenamiento de archivos (incluida la colaboración en archivos) e integración de aplicaciones. El servicio se integra con el paquete de productividad de Office por suscripción y presenta extensiones que pueden integrarse con productos que no son de Microsoft.

[6] Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Figura 5

Repositorio empleado para el trabajo fin de máster



2. Contexto y estado del arte

En este capítulo se describen las técnicas de aprendizaje automático más utilizadas para la predicción en medicina en el ámbito que abarca el presente trabajo:

- Anonimización de datos[7] en el ámbito de la salud.
- Construcción de datasets y modelos predictivos que ayuden a optimizar la gestión hospitalaria.

Con el crecimiento exponencial de los datos médicos depositados en las bases de datos, se necesitan modelos adecuados para extraer de ellos datos significativos. El objetivo del aprendizaje automático es "*optimizar un criterio de rendimiento basado en la experiencia previa*" [(Larrañaga P y otros, 2006)]. Se emplea la teoría estadística para crear modelos basados en predicciones realizadas a partir de resultados anteriores. El reconocimiento de patrones se refiere al descubrimiento de regularidades en la información mediante algoritmos informáticos y la posterior clasificación de los datos en categorías [(Bishop, C. M. & Nasrabadi, N. M., 2006)]. Para ello trata de utilizar el procesamiento de la información

[7] La anonimización de datos define la metodología y el conjunto de buenas prácticas y técnicas que reducen el riesgo de identificación de personas, la irreversibilidad del proceso de anonimización y la auditoría de la explotación de los datos anonimizados, monitorizando quién, cuándo y para qué se usan. Es decir, cubre tanto el objetivo de anonimización, como el de mitigación del riesgo de reidentificación, siendo este último un aspecto clave.

para resolver problemas que abarcan una gran variedad de temas, como el reconocimiento del habla, la clasificación de caracteres manuscritos y el diagnóstico médico. Los humanos tienden a resolverlo sin esfuerzo, pero su aplicación en los ordenadores ha supuesto un reto [(Bishop, 1995)]. En este capítulo se describen las técnicas de aprendizaje automático más utilizadas para la predicción en medicina.

En el presente apartado se estudiarán algunos casos de uso desarrollados por otros autores y los modelos predictivos que se utilizarán a lo largo del trabajo.

La predicción básica se ha utilizado en diversos campos. Por ejemplo, Nate Silver, fundador y editor jefe de FiveThirtyEight, es un estadístico que utiliza modelos de predicción con el fin de predecir las elecciones estadounidenses con un alto grado de éxito gracias a la recopilación y el análisis de datos políticos para dar el posible resultado de las próximas elecciones [(Silver, 2008)].

Otro ejemplo es el de Google, que descubre las tendencias y las fluctuaciones del mercado bursátil [(Leinweber, 2013)] al encontrar la correlación entre las búsquedas en Internet del nombre de una empresa y su volumen de negocio, aunque no pudo predecir su precio en la bolsa.

En el campo de la medicina, en los años 60, se empezaron a utilizar sistemas de ayuda a la decisión de diagnósticos con aplicaciones bayesianas. Se propusieron varios métodos para eliminar el efecto de los resultados redundantes [(Miller, 1977)].

En la década de los 90, varios trabajos analizaron las técnicas de aprendizaje automático supervisadas y no supervisadas [(Hadzikadic, 1992)] aplicadas a la medicina. Un ejemplo de técnica supervisada es la red neuronal, formada por "elementos de procesamiento activo cuyas interacciones locales a lo largo del tiempo conducen al comportamiento global del modelo" [(Reggia, 1993)]. En [(Baxt , 1992)], Baxt utiliza las redes neuronales para identificar patrones y relaciones complejas que son difíciles de encontrar por un humano. La aplicación de redes neuronales para identificar el infarto agudo de miocardio en pacientes que acuden a urgencias cuando tienen dolor torácico anterior, ha demostrado ser más precisa que los médicos.

La heurística también se ha utilizado como análisis predictivo, ayudando a los médicos a tomar mejores decisiones. Con la heurística, la estrategia consiste en ignorar parte de la información y centrarse en un par de elementos clave para participar en la predicción. Marewksi y Gigerenzer [(Marewski & Gigerenzer, 2012)] explican cuándo la heurística puede superar a otros métodos que utilizan mucha más información. El método consiste en

hacer algunas preguntas de sí o no, como simplemente buscar una anomalía en el electrocardiograma del paciente, o por ejemplo si el paciente se queja de dolor en el pecho. Dentro de la misma línea de investigación, se creó una regla de decisión clínica muy sensible para decidir si se debe realizar o no una tomografía computarizada (TC) a los pacientes con traumatismos craneoencefálicos leves, debido a que sólo un pequeño porcentaje de pacientes empeora y requiere intervención médica. En ese caso, es necesario realizar un diagnóstico precoz del hematoma intracraneal mediante TC y la posterior intervención quirúrgica.

En [(Kline y otros, 2008)] se utilizaron reglas de decisión para decidir si debía realizarse una prueba de embolia pulmonar (EP) en pacientes con factores de riesgo bajos, ya que los expertos sugieren que la EP sigue pasándose por alto en un alto porcentaje.

Otro uso popular del análisis predictivo es el desarrollo de métodos de aprendizaje automático para predecir la mortalidad en pacientes diagnosticados inicialmente de neumonía. Esto es útil para dar al médico una idea de si el paciente es apto para irse a casa o debe permanecer en el hospital para recibir más cuidados intensivos [(Cooper, y otros, 1997)].

2.1 Casos de uso similares

A continuación, se detallan algunos casos de uso en el ámbito de la inteligencia artificial en entornos de salud para la creación y validación de modelos predictivos, comenzando con la anonimización de datos. Se entiende caso de uso como un trabajo con una orientación vertical a la resolución de un problema concreto en un contexto concreto. Con el presente apartado se pretende detallar de qué datos parten, qué algoritmos utilizan y que resultados obtienen y en base a qué métricas.

2.1.1 Anonimización de datos

El trabajo de anonimización de datos (Alekh, 2015) se centra en los protocolos DICOM (digital imaging and communications in medicine) y HL7 (health level seven), estándares de comunicaciones para la transferencia de datos de imágenes y texto respectivamente entre aplicaciones en el ámbito sanitario basado en el estándar HIPAA (equivalente estadounidense al europeo GDPR) identifica los datos que deben ser cifrados o eliminados en el ámbito de salud. En el marco de los datos de texto incluido en protocolo HL7 habla sobre la pseudo-anonimización y la anonimización de la información detallando que la

pseudo-anonimización es aquella que permite asignar un pseudónimo a cada paciente con el objetivo de poder identificarlos a través de este sin que se conozca su identidad.

Por otro lado, en el proceso de anonimización se eliminan todos los datos que permitirían identificar a un paciente concreto. Estos datos son los siguientes.

1. Nombres
2. Datos geográficos
3. Fechas
4. Datos de contacto
5. Números de registros médicos
6. Números de seguros médicos
7. Números de cuenta
8. Números de certificados y licencias
9. Identificadores de vehículos
10. Números de serie
11. URLs e IPs
12. Datos biométricos
13. Imágenes de caras

No hemos encontrado estudios ni trabajos aplicados al equivalente europeo de la normativa HIPAA, el Reglamento General de Protección de Datos (RGPD, 2016).

2.1.2 Obtención de variables de mensajes HL7 para la predicción de readmisión a 30 días

La readmisión hospitalaria no planificada representa grandes costes al sector sanitario. Los autores (Swain & Kharrazi, 2015) describen cómo han identificado diferentes modelos de riesgo de re-ingreso hospitalario (Readmission Risk Prediction Models) y han estudiado las variables que se pueden extraer del conjunto de mensajería HL7 para alimentar estos modelos.

El estudio parte de trabajos publicados antes de 2013, extrayendo las variables utilizadas para la generación de los modelos y las buscan en el contenido de mensajes HL7. Identifican 297 variables predictivas tras la revisión de 32 artículos.

El trabajo consigue obtener el total de las variables extrayéndolas de los segmentos:

- DG1 (Diagnóstico): 89.2%
- PID (Identificación): 13.9%
- OBX (Observación): 9.1%

La publicación concluye que se pueden extraer datos mediante el procesado de mensajería HL7 siempre y cuando la integridad de estos datos supere ciertos umbrales.

Por otro lado, el trabajo *Predicting 30-day Hospital Readmission with Publicly Available Administrative Database* (Zhu y otros, 2015)*. A Conditional Logistic Regression Modeling Approach* utiliza “Condicional Logistic Regression” (CLR) y validación cruzada. El trabajo mejora resultados en comparación con algoritmos como “regresión logística directa”, “regresión logística paso a paso”, “random forests”, “Support Vector Machine”.

Utiliza variables predictoras identificadas a partir de los datos de los datos del HCUP (Healthcare Cost and Utilization Project, de California) que incluyen la ubicación desde el alta, la cantidad de afecciones crónicas y la cantidad de procedimientos agudos. El trabajo concluye que los modelos CLR puede ayudar a desarrollar a futuro modelos de predicción para identificar el riesgo de reingreso y subraya la importancia de la recopilación de datos sobre otros marcadores y su estructuración en bases de datos para poder desarrollar más estudios.

2.1.3 Predicción del uso frecuente de urgencias mediante Predicción logística multivariable

El trabajo (Wu y otros, 2016) desarrolla un modelo basado en regresiones multivariable que permite clasificar al paciente de urgencias como “frecuente” cuando tiene más de 4 visitas anuales. Este tipo de pacientes representa entre el 4.5% y el 8%.

Utiliza variables demográficas como edad, sexo. También utiliza el número de visitas durante un periodo (2018), así como “Chief Complaint”, que determina la categoría de las de la dolencia en base a un catálogo determinado. Otra de las variables (Figura 6) que utiliza es

la distancia que calcula a través del código postal de origen del paciente y código postal del centro

Figura 6

Distribución de visitas a urgencias por distancia de viaje y quejas en cada año, de 2008 a 2010

Characteristics	2008 (%)	2009 (%)	2010 (%)
Travel distance (miles)			
<=5	60.1	60	60.1
5–20	30.2	30.7	30.3
>20	8.4	8.1	8.3
Missing	1.4	1.2	1.3
Chief complaint[a]			
Respiratory	23.1	24.6	22.0
Gastrointestinal	17.1	17.6	17.7
Neurologic	11.3	11.2	11.2
Skin	4.6	4.4	4.7
UDI	9	10.5	8.8
Lymphatic	3	3.0	3.0
ILI	18.8	20.7	20.9
Dental	1.8	1.8	1.8
Pain	41.5	40.3	40.8
Musculoskeletal	28.7	27.5	27.7
Alcohol	0.5	0.5	0.6
Unclassified	15.7	14.8	15.4
Missing	1.34	1.47	3.41

Fuente: (Wu y otros, 2016)

Parte de datos HL7 (V2) de 96 instituciones participantes en el PHESS (Public Health Emergency Surveillance System) entre enero de 2008 y diciembre de 2010, preprocesados con los que se genearon datasets válidos. Incluye referencias a datos de segmentos concretos del segmento Patient Visit (PV1) y Patitent Identification (PID). Optan por excluir registros con datos faltantes y por la agrupación de episodios pertenecientes a un mismo paciente asignando un identificador de global único que les ayuda a emparejar todas las visitas de un paciente en diferentes bases de datos.

El artículo concluye que existe correlación entre las variables predictoras y la frecuencia de uso de los servicios de urgencia con resultados AUC (Area under the curve) entre 0.84 y 0.92. Determina que es técnicamente factible utilizar datos de registro para predecir este comportamiento y validarán el algoritmo con otros centros de salud de los EEUU.

2.1.4 Predicciones en triaje

En el estudio de (Raita, 2019) Utilizan LASSO regression (least absolute shrinkage and selection operator), random forest, gradient boosted decision tree, y redes neuronales para priorizar pacientes en el momento del triaje a partir de datos demográficos, información de triaje, y datos de morbilidad asociada, con el objetivo de saber si van a necesitar acudir a una unidad de atención crítica (en la que se incluyen los casos de muerte intra-hospitalaria). El objetivo del trabajo es ayudar a asignar eficientemente los recursos hospitalarios, y priorizar pacientes de alto riesgo.

Parten de datos de adultos (> 18 años) de visitas registradas entre 2007 y 2015 del National Hospital and Ambulatory Medical Care Survey (NHAMCS). Para realizar el estudio separan el 70% de los datos en un dataset de entrenamiento y utilizan el 30% restante para medir los resultados.

Obtienen resultados clasificando pacientes que necesitarán cuidados intensivos e ingreso hospitalario, obteniendo resultados AUC de 0.86 utilizando redes neuronales para cuidados intensivos y AUC de 0.82 para hospitalización con técnicas de machine learning. En el caso de hospitalización, obtienen peores resultados con redes neuronales.

El artículo detalla variables importantes (Figura 7) que utiliza, pero no indica de qué segmentos HL7 obtiene los datos:

- Edad del paciente
- Sexo
- Datos clínicos de triaje: temperatura, presión sanguínea, frecuencia respiratoria, etc.
- Morbilidad asociada

Figura 7

Variables predictoras y resultados en 135.470 visitas al servicio de urgencias de adultos

Variable
Age (year), median (IQR)
Female sex
Mode of arrival
Ambulance
Emergency Severity Index
1 (immediate)
2 (emergent)
3 (urgent)
4 (semi-urgent)
5 (non-urgent)
Vital signs
Temperature (F), median (IQR)
Pulse rate (bpm), median (IQR)
Systolic blood pressure (mmHg), standard deviation (SD)
Diastolic blood pressure (mmHg), standard deviation (SD)
Respiratory rate (per min), median (IQR)
Oxygen saturation (%), median (IQR)
Common chief complaints
Musculoskeletal-related complaints
Gastrointestinal-related complaints
General complaints (e.g., fever)
Injuries
Respiratory-related complaints
Neurological-related complaints
Urological-related complaints
Psychiatry-related complaints
Treatment-related complaints (e.g., side effects)
Eye and ear-related complaints
Skin-related complaints
Intoxication
Elixhauser comorbidity measures (≥ 1)
Clinical outcomes
Critical care outcome*
Hospitalization outcome†

Fuente: (Raita, 2019)

2.1.5 Predicción de mortalidad a un año

El estudio (Sahni, 2018) se centra en realizar una prueba de concepto para determinar si es posible predecir el riesgo de mortalidad a un año en el momento del alta hospitalaria.

Parte de un dataset creado con la historia clínica de 59.848 pacientes en una red de 6 hospitales del área urbana de Minnesota durante un periodo de 4 años. El estudio emplea cuatro clases de variables (obtenidas en las 48h anteriores al alta), es decir, utiliza la última foto de la información disponible de cada paciente antes de su alta incluyendo:

- Variables demográficas: duración de estancia, sexo, edad.
- Variables fisiológicas: presión sanguínea, temperatura, pulso, frecuencia respiratoria, etc.
- Variables biomédicas: niveles de potasio, sodio, creatina, etc.
- Variables de comorbilidad.

Es importante destacar que realizan diferentes pruebas para completar los datos faltantes basadas en estrategia de vecinos y cálculo de mediana obteniendo los mismos resultados. Dado que el rendimiento de los modelos era similar, los autores optan definitivamente por el cálculo de la mediana ya que es más rápido y sencillo a nivel computacional.

Aunque el trabajo no parte directamente de datos extraídos de mensajería HL7, indica que las variables utilizadas pueden extraerse de este estándar, siempre y cuando el sistema de historia clínica propague esta información, y que de esta forma se podría desarrollar un sistema automatizado de toma de decisiones.

A nivel de algoritmos, utiliza Random Forests y regresiones logísticas obteniendo un AUC de entre el 0.85 y 0.87, indicando en sus resultados que en grupos de pacientes con alta probabilidad de muerte teniendo menos de un 10% de falsos negativos.

Los autores concluyen que el uso de Random Forests ayuda a trabajar con este tipo de datasets con mucho ruido concluyendo además que las variables fisiológicas y bioquímicas son las más predictivas.

2.1.6 Predicción a corto plazo de ingresos en urgencia

El estudio *Machine Learning for Real-Time Aggregated Prediction of Hospital Admission for Emergency Patients* (King, y otros, 2022), utiliza técnicas de aprendizaje automático para predecir los ingresos a corto plazo en el servicio de urgencias del Hospital Universitario del Reino Unido.

Está basado en datos del registro electrónico de pacientes (sistema de historia clínica). Al igual que el presente trabajo, parten de datos generados por mensajería HL7 que es

almacenada en una base de datos relacional PostgresSQL[8], La base de datos incluye el registro completo del paciente, incluidas las observaciones, resultados de las patologías, la ubicación de los pacientes, las solicitudes de consulta y un resumen del historial de visitas anteriores. A diferencia del estudio presente, parten de un esquema relacional ya creado con el que normalizan datasets utilizando **R.**

Sugiere que se puede mejorar los servicios hospitalarios si se pronostica a corto plazo el ingreso en pacientes de urgencia, ya que parte de estos derivarán a hospitalización, por lo que ayudaría a planificar ciertos recursos. Habla de estudios anteriores en los que se han empleado desde datos meteorológicos, de salud pública y geográficos destacando que además en hospitales donde se utiliza registro electrónico se puede partir también de estos datos para predecir la demanda de camas.

El trabajo utiliza un sistema de predicción que se ejecuta 4 veces al día en ventanas temporales de entre 4 y 8 horas. Realizan las predicciones basadas en modelos probabilísticos en 7 pasos.

Utiliza clasificadores XGBoost aplicado a 109465 visitas para obtener un AUROC de entre 0.82 y 0.90.

2.2 Modelos predictivos

2.2.1 Modelo k-Nearest Neighbour

El modelo de k-Nearest Neighbour es un método utilizado para la clasificación, el reconocimiento de patrones y la regresión. k-Nearest Neighbour es un algoritmo de aprendizaje supervisado. Funciona bien cuando cada límite de decisión es muy irregular, como los dígitos escritos a mano o las imágenes por satélite. Este clasificador utiliza los propios casos para clasificar las instancias y no necesita ajustarse a un modelo [(Cooper y otros, 1997)]. Se considera un algoritmo de aprendizaje basado en la memoria porque acumula las instancias de entrenamiento en una tabla de búsqueda e interpola a partir de ellas [(Islam, M. 2007)]. Suelen ser los que mejor se comportan para las soluciones de la vida real, aunque se comportan mal para los problemas de alta dimensión en la clasificación [(John Lu, 2010)]. Se basa en el principio de que las instancias dentro de una base de datos con propiedades similares estarán cerca unas de otras. Las instancias se consideran como puntos dentro de un espacio n-dimensional en el que cada n-dimensión se correlaciona con

[8] PostgreSQL, también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL similar a la BSD o la MIT.

una n-característica utilizada para caracterizar una instancia. La distancia relativa entre estos puntos importa más que su posición absoluta en el espacio [(Maglogiannis, I]. k-Nearest Neighbour utiliza el método del prototipo para representar los datos de entrenamiento en puntos del espacio de características. Este prototipo tiene una etiqueta de clase asociada. El algoritmo debe averiguar cuántos prototipos utilizar y dónde aplicarlos. Hay muchas formas de calcular la distancia entre instancias, como la distancia de Manhattan, la de Minkowsky o la de Chebychev, pero el algoritmo suele utilizar la distancia euclidiana en el espacio de características para determinar cuál es el prototipo específico más cercano a un punto x. Dado el punto de consulta x, el algoritmo encuentra los k puntos de entrenamiento más cercanos a x y, posteriormente, clasifica en función de la mayoría de votos otorgados por los k vecinos [(John Lu, 2010)]. En otras palabras, encuentra el número de k casos que son más similares al nuevo caso. Es una forma de razonamiento basado en la atención [(Cooper y otros, 1997)]. Cada característica se convierte para que tenga media cero y varianza 1, por si acaso cada característica toma unidades diferentes. El rendimiento del algoritmo varía en función del número de valores k asignados. No existe un método de principio para elegir el valor de k. Un método común para elegir el valor de k es la validación cruzada o probar múltiples valores de k y calcular el error de clasificación y elegir el k que proporciona el menor error [(Maglogiannis, I]. Los números impares de k se utilizan habitualmente para romper los empates (1,3,5,7 o 9). Cuanto mayor sean los valores de k, más ayudan a reducir las consecuencias del ruido dentro del conjunto de datos de entrenamiento, aunque hay que tener cuidado con la elección de valores grandes de k porque tienden a clasificar mal si las clases individuales no están muy separadas [(Islam, M. 2007)].

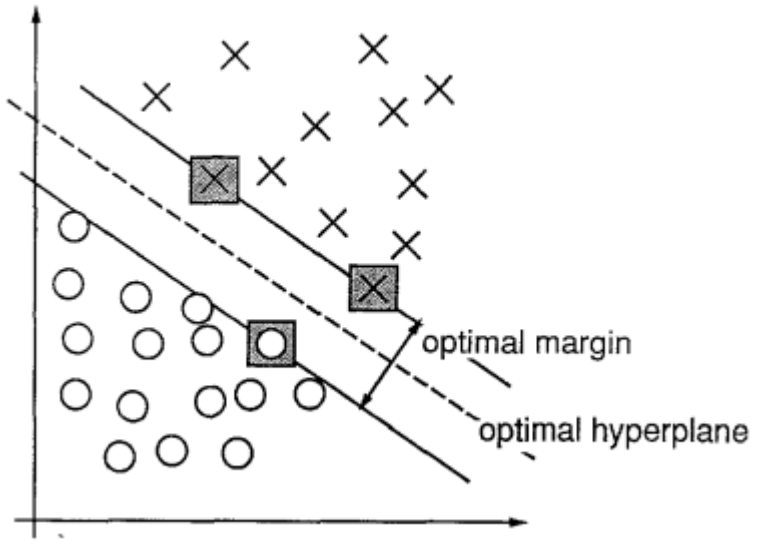
2.2.2 Support Vector Machines

Las Support Vector Machines (SVM) se hicieron populares para resolver problemas de clasificación, regresión y detección de novedades [(Bishop, C. M. & Nasrabadi, N. M., 2006)]. El funcionamiento de las SVM consiste en procesar los datos para obtener una dimensión superior a la que tenía el conjunto original para poder separar en dos categorías diferentes con un hiperplano [(Larrañaga P y otros, 2006)]. Es en este espacio de alta dimensión donde se realiza una superficie de decisión lineal debido a sus propiedades especiales que le permiten crear un buen modelo generalizador [(Cortes, C. & Vapnik, V., 1995)]. Para hacer este modelo óptimo y conseguir un mejor clasificador, el algoritmo debe encontrar el hiperplano con mayor margen (ver Figura 8). Para hacer este hiperplano óptimo, sólo se necesita una pequeña cantidad de datos de entrenamiento, que se llaman vectores de soporte, para decidir este margen [(Cortes, C. & Vapnik, V., 1995)]. Una ventaja

de las SVM es que la función objetivo es convexa, por lo que la solución de la optimización es sencilla [(Bishop, C. M. & Nasrabadi, N. M., 2006)]. Las SVM es básicamente un clasificador de dos clases, aunque en la práctica solemos tener problemas con más de dos clases. Es posible combinar múltiples SVM de dos clases para crear un clasificador SVM multiclase.

Figura 8

Un ejemplo de problema separable en un espacio bidimensional. Los vectores de soporte, marcados con cuadrados grises, definen el margen de mayor separación entre las dos clases



Fuente: (Cortes, C. & Vapnik, V., 1995))

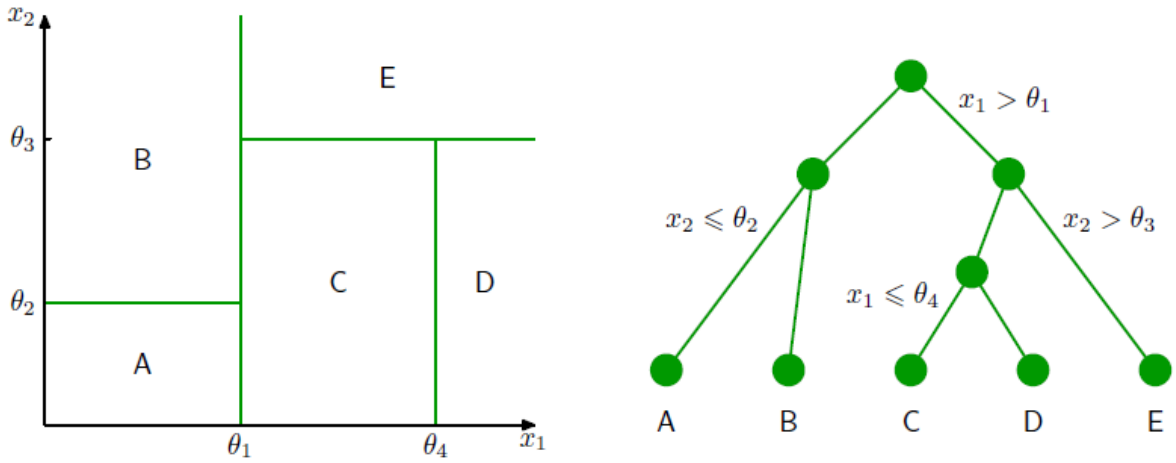
2.2.3 Árboles de clasificación y regresión (CART)

Los modelos basados en árboles son un método combinatorio en el que sólo un modelo realiza predicciones en cualquier punto del espacio de entrada. Se divide el espacio en rectángulos y se coloca un modelo en cada uno de ellos [(John Lu, 2010)]. Para crear el modelo, dada una entrada, se realiza un proceso de elaboración secuencial en el que el árbol se divide en dos ramas en cada nodo. Se trata de una secuencia de decisiones binarias a variables de entrada únicas [(Bishop, C. M. & Nasrabadi, N. M., 2006)]. En la Figura 9, podemos ver cómo la primera entrada se divide en dos nodos diferentes dependiendo de si el parámetro del modelo (θ) es mayor o no que X1. Además, se puede ver que $X1 < \theta_1$ se divide a su vez dependiendo de si X2 es menor o igual que θ_2, lo que forma dos nuevas regiones: A y B. Así, para cualquier nueva entrada, el árbol decidirá en qué región cae, gracias a los criterios de decisión de cada nodo. La razón por la que es importante en la investigación médica se debe a su simplicidad y a que es fácilmente

interpretable. El espacio de características se explica completamente con un solo árbol [(John Lu, 2010)].

Figura 9

Árbol binario con su correspondiente partición del espacio de entrada



Fuente: (Bishop, C. M. & Nasrabadi, N. M., 2006))

Los factores clave para crear un modelo de árbol de regresión son la determinación de las variables que se utilizarán para tomar la decisión binaria, así como la estructura del árbol y el umbral de cada nodo. Suponiendo que el modelo se dividirá en M regiones R1, R2, …, RM, y se modela la respuesta como una constante c_m en cada región:

Debido a la gran cantidad de soluciones combinatorias, la estructura del árbol se crea formando un nodo individual cada vez, y la elección de las variables a dividir y el valor de cada umbral se determina mediante una búsqueda exhaustiva. Esta búsqueda encontrará una variable y un umbral óptimos con respecto a la media local de los datos, que dará el menor error de la suma de los cuadrados [(Bishop, C. M. & Nasrabadi, N. M., 2006)].

Si aplicamos el criterio de minimizar el error de la suma de los cuadrados, el óptimo $\hat{c}_m$ es la media de y_i en la región Rm.

Para cada variable de división se determina un punto de división s y, buscando en todas las entradas, se encuentra el par óptimo (j, s). Una vez encontrada la mejor división, los datos se dividen en dos regiones y el proceso de división se realiza de nuevo en cada una de las dos regiones siguientes. Este proceso se repite en todas las regiones [(John Lu, 2010)]. Después, el algoritmo debe decidir el tamaño del árbol. Un árbol grande sobreajustará los datos, mientras que un árbol demasiado pequeño no definirá la estructura importante. El

tamaño del árbol determinará la complejidad del modelo, y debe elegirse en función de los datos. Un enfoque para determinar el tamaño del árbol consiste en dividir los nodos del árbol si la disminución de la suma de los cuadrados supera algún umbral. El problema de este enfoque es que una división que parece inútil puede conducir a una división aún mejor por debajo de ella. Otro enfoque consiste en encontrar un árbol más grande “T1” y detener la división cuando se alcanza un determinado tamaño de nodos. Después, se puede podar el árbol mediante un método llamado poda de complejidad de costes. Este método consiste en definir un árbol T obtenido mediante la poda de “T1”. Los nodos terminales están indexados por m, y Rm representa el nodo m, y "T" define el número de nodos terminales en T.

Seguidamente, para cada α, se encuentra el subárbol T_α que minimiza C_α (T).

El parámetro de ajuste $\alpha \geq 0$ determina el equilibrio entre el tamaño del árbol y la forma en que se ajusta a los datos. Los valores grandes de α darán árboles más pequeños T_α. Si $\alpha = 0$, se dará el árbol completo “T1”. Cada α dará un único T_α que minimiza C_α. Para calcular α se utiliza una validación cruzada de tamaño cinco o diez, y se elige un α que minimiza la suma de cuadrados de la validación cruzada. El árbol final será $T_{\hat{\alpha}}$ [(John Lu, 2010)].

Una de las desventajas del modelo de regresión en árbol es que es muy sensible a pequeños cambios en los datos de entrenamiento [(Bishop, C. M. & Nasrabadi, N. M., 2006)]. Cuando se realiza un pequeño cambio, éste conduce a un conjunto de divisiones muy diferente [(John Lu, 2010)] Esto se debe al proceso jerárquico del modelo, un error en la división superior conducirá a errores en las divisiones inferiores.

2.2.4 Técnicas basadas en árboles

Estas desventajas de los modelos de árboles no han podido solucionarse mejorando los algoritmos o minimizando los errores de las funciones. Sin embargo, se han creado otras técnicas que se basan en la combinación del resultado de varios árboles. Esta técnica de combinación se conoce con el nombre de ensamblado o ensemble learning. Los métodos ensemble [(Koren, 2009); (Dietterich & et al, 2002)] consisten en la construcción de predicciones a partir de la combinación de las predicciones de varios modelos. De esta forma, se comprueba si realizando dicha combinación puede producir una mejora respecto a los resultados discretos.

Estos métodos no solo se basan en combinar distintas técnicas o algoritmos de predicción, como los que ya hemos comentado y otros que comentaremos más adelante, sino que pueden realizar el ensamblado utilizando un mismo tipo de predictor. Dentro de ellas

encontramos las técnicas basadas en la generación de estructuras arborescentes como las que veremos a continuación.

2.2.4.1. Bagging

Según [(Breiman L. , 1996)] la técnica bagging, diminutivo de bootstrap aggregation, consiste en crear diferentes modelos usando muestras aleatorias con reemplazo y luego combinar o ensamblar los resultados. Esta técnica sigue estos pasos:

1. Se selecciona el conjunto de datos de entrenamiento y se divide en distintos subconjuntos de datos con o sin reemplazamiento.
2. Con cada una de estas submuestras obtenidas se crea un modelo predictivo, obteniendo por tanto modelos diferentes.
3. Se construye un único modelo predictivo, que es el resultado de promediar todos los modelos creados.

La ventaja de utilizar la técnica bagging es que no hay dependencia de los datos y se evita la inestabilidad generada por un árbol simple. Como consecuencia, se reduce la varianza. Además, se suavizan las predicciones obteniendo valores más sutiles y adaptados, ya que no se aplicará el mismo valor predictivo como ocurre en los árboles de decisión.

No obstante, tenemos una desventaja, y es que en esta técnica se usan siempre las mismas variables para construir todos los árboles. Por lo que, si estamos en presencia de una variable dominante, la mayoría de los modelos generados serán iguales o muy parecidos.

Es importante destacar que en el proceso de bagging, el número de árboles generados no es un parámetro por el que debemos preocuparnos. Esto es así ya que por mucho que se incremente el número de unidades, no se aumenta el riesgo de sobreajuste del modelo. Cuando se alcanzada un determinado número de árboles, el valor del error se estabiliza.

2.2.4.2. Random Forest

Esta técnica [(Breiman L. , 2001)] es una extensión de bagging, o dicho de otra manera: bagging es un caso particular de random forest. Consiste en incorporar aleatoriedad en las variables utilizadas para segmentar cada nodo del árbol. De forma análoga al modelo anterior, describiremos los pasos que sigue esta técnica:

1. Se selecciona el conjunto de datos de entrenamiento y se divide en distintos subconjuntos de datos con o sin reemplazamiento.
2. En cada nodo, seleccionamos subgrupos de variables entre todas las originales y de ese subgrupo de variables, se escoge la mejor para la partición del nodo.

3. Con cada una de estas submuestras obtenidas se crea un modelo predictivo, obteniendo modelos diferentes.
4. Se construye un único modelo predictivo, que es el resultado de promediar todos los modelos creados.

Por tanto, la ventaja de esta técnica es que se evita que haya variables dominantes y se aprovecha mejor la información del resto de variables del conjunto. De esta forma nos aseguramos de que los árboles puedan ser muy diferentes y capten sutilezas importantes, evitando así problemas de sobreajuste.

Además, tal y como ocurre en bagging, tampoco hay que tener en cuenta el número de árboles generado, ya que llegados a un punto el error se estabiliza.

2.2.4.3. Gradient boosting

A diferencia de los dos métodos anteriores, consiste en un algoritmo iterativo. La idea es ajustar, de forma secuencial, varios modelos de árboles.

Con cada modelo que se genera se utiliza la información del modelo anterior para aprender de sus errores. De esta forma se intenta mejorar el resultado en cada iteración.

Tanto en bagging como en random forest la cantidad de árboles generado no es un parámetro que pueda afectar a los modelos. Sin embargo, en boosting, si creamos demasiados árboles, puede conllevar a sobreajuste. Para evitar este problema [(Friedman, 2001)] podemos ajustar un parámetro llamado learning rate o shrinkage (tasa de aprendizaje) que gradúa cuánto se va corrigiendo el error en cada iteración. Hará falta un número mayor de iteraciones (árboles) cuando se introduce un valor de shrinkage muy pequeño, mientras que para tasas moderadas el número de iteraciones necesarias es inferior.

2.2.5 Naive Bayes

En un sentido amplio, los modelos de Naive Bayes son una clase especial de algoritmos de clasificación de Aprendizaje Automático, o Machine Learning. Se basan en una técnica de clasificación estadística llamada “teorema de Bayes” (Morales & Escalante, 2012).

Estos modelos son llamados algoritmos “Naive”, o “Inocentes” en español. En ellos se asume que las variables predictoras son independientes entre sí. En otras palabras, que la presencia de una cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de cualquier otra característica.

Los puntos fuertes principales son:

- Un manera fácil y rápida de predecir clases, para problemas de clasificación binarios y multiclase.
- En los casos en que sea apropiada una presunción de independencia, el algoritmo se comporta mejor que otros modelos de clasificación, incluso con menos datos de entrenamiento.
- Funciona bien en el caso de variables de entrada categóricas comparada con variables numéricas.

Los puntos débiles principales son:

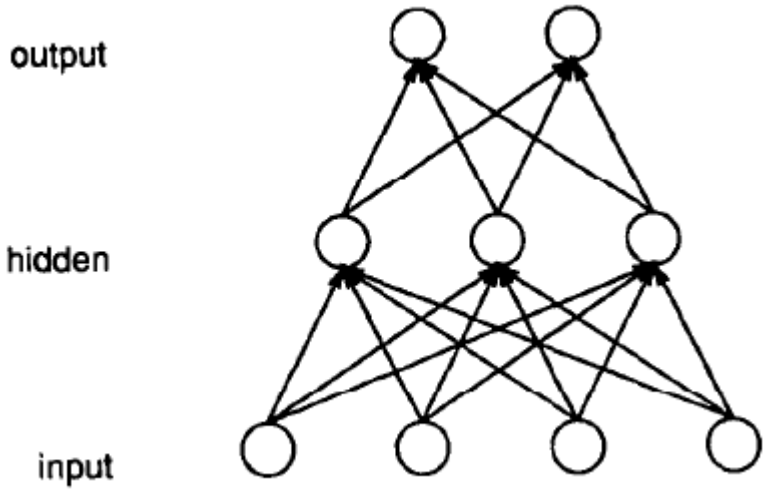
- La presunción de independencia Naive muy probablemente no reflejará cómo son los datos en el mundo real.
- Si la variable categórica tiene una categoría en el conjunto de datos de prueba, que no se observó en el conjunto de datos de entrenamiento, el modelo asignará una probabilidad de 0 y no podrá hacer una predicción. Esto se conoce a menudo como frecuencia cero. Uno de los principales métodos para evitar esto, es la técnica de suavizado, siendo la estimación de Laplace una de las más populares.

2.2.6 Redes neuronales

La creación de redes neuronales fue motivada históricamente por las neuronas interconectadas en el cerebro [(Reggia, 1993)]. Puede definirse como una red de elementos de procesamiento que conducen a un comportamiento global del modelo (ver Figura 10). Las redes neuronales crean combinaciones lineales de las entradas y crean un modelo como una función no lineal de estas características [(John Lu, 2010)]. Puede utilizarse tanto como modelo de regresión como de clasificación. Las unidades elementales de procesamiento se denominan neuronas o nodos y se dividen en capas de forma que sólo se conectan las unidades situadas dentro de dos capas consecutivas [(Larrañaga P y otros, 2006)]. La red neuronal más sencilla se llama perceptrón. Consiste en un clasificador de un neutrón que utiliza una función de activación de umbral para separar dos clases mediante una función de discriminación lineal.

Figura 10

Red neuronal con cuatro nodos de entrada, tres nodos ocultos y dos nodos de salida. Las conexiones van en la dirección de la entrada a los nodos ocultos y de los nodos ocultos a los nodos de salida



Fuente: [(Reggia, 1993)]

La combinación de varios perceptrones se denomina perceptrón multicapa. La salida de las capas intermedias se envía sólo a la capa más alta. Un perceptrón multicapa con sólo dos capas ocultas puede aproximarse a cualquier problema de clasificación [(Larrañaga P y otros, 2006)].

En las redes neuronales, cada nodo n_i está asociado a un nivel de activación numérico $a_i(t)$ y a un tiempo t [(Reggia, 1993)]. Un nodo comunica su nivel de activación a su entorno en cualquier momento. Los nodos reciben la activación total de entrada $in_j(t)$ que utilizan para actualizar su propio nivel de activación. Además, cada conexión está asociada a un peso w_{ji} y también se utiliza para actualizar las activaciones de los nodos. Si una activación de un nodo n_i tiende a aumentar la activación del nodo vecino n_j, entonces la conexión se denomina enlace excitador y se etiqueta con $w_{ji} > 0$. Por otro lado, si una activación de un nodo n_i tiende a disminuir la activación del nodo vecino n_j, entonces la conexión es un enlace inhibidor y se etiqueta con $w_{ji} < 0$.

Otro componente de las redes neuronales es la regla de aprendizaje, que se refiere a las modificaciones en la red debido a sus experiencias a lo largo del tiempo. Esta regla de aprendizaje describe cómo cambian los pesos de las conexiones en función del tiempo.

2.3 Aportación del trabajo

La revisión de la bibliografía revela que existen trabajos relacionados con los algoritmos predictivos en el ámbito de la medicina. Tal y como se describe en los apartados anteriores, existen relaciones con los tres aspectos que aborda el proyecto:

1. Anonimización previa de los datos: Se parte del trabajo de [(Alekh, Sunder, & Gaur, 2015)] para identificar técnicas de anonimización de datos HL7 basados en la normativa HEPAA. En el caso del presente trabajo se desarrollarán técnicas para el procesado de la mensajería basados en tecnologías .NET que nos facilitarán la generación de grandes estructuras de datos y no nos ayudarán a eliminar en un primer paso los datos identificativos más relevantes procesando grandes volúmenes de información, para poder generar datasets totalmente anonimizados en un segundo paso, cumpliendo con los estándares citados en el artículo.

2. Generación de datasets: Los trabajos analizados que utilizan mensajería HL7 parten de información fuertemente relacionada con segmentos de información que albergan datos clínicos: DG1 (Diagnóstico) y OBX (Observaciones). El presente trabajo trata de aportar en este ámbito un enfoque diferente, como por ejemplo el uso de información relacionada con los movimientos del paciente a través del hospital, para generar datasets con información referente a la estancia en ciertas unidades de enfermería, así como información relacionada con posibles reingresos, posibilidad de muerte, o necesidades de recursos específicos. El trabajo se centra por lo tanto en generar predicciones en base a la información contenida en los segmentos (ADT e ITR) que pudiera aportar variables relevantes para aplicar a ciertos casos de uso.

   Por otro lado, cabe destacar que otros trabajos parten de bases de datos estructuradas en forma de **datasets planos**. El presente estudio no implementa modelos basados en datasets ya creados, sino que tiene un importante componente en el ámbito de la minería de datos, partiendo de una secuencia de mensajes HL7 e implementando un procesador que mapea y estructura esta secuencia en datasets que luego serán utilizados para entrenar y testear modelos.

   Como se ha comentado, cada mensaje recibido está relacionado con una acción sobre un episodio. Una vez anonimizado el mensaje se va construyendo cada registro del dataset en base al procesamiento secuencial de todos los mensajes.

3. Modelos: La mayoría de los estudios o experimentos que utilizan esta tipología de datos, centra más su atención en el algoritmo a elegir, que en el resto de las acciones que hay que llevar a cabo. En el presente trabajo, además de realizar varios modelos predictivos, se destaca las técnicas realizadas para tratar un conjunto de datos que cuenta con un gran porcentaje de valores ausentes y tiene un gran desbalanceo entre las clases a predecir.

Se considera una aportación el desarrollo de un software que sea capaz de llevar a cabo esas predicciones de manera adecuada, ya que puede suponer el primer paso para el

desarrollo de una herramienta que facilite la gestión eficiente de los recursos en el ámbito sanitario.

3. Objetivos y metodología de trabajo

En este capítulo se detallan los objetivos tanto generales como específicos que se pretenden alcanzar con este trabajo.

Más adelante se expondrá la metodología que se ha seguido para poder cumplir con los objetivos propuestos.

3.1 Objetivo general

El objetivo general de este proyecto es conseguir desarrollar un software que implemente modelos que permitan realizar ciertas predicciones en ciertos puntos del **circuito** habitual de un paciente en un centro hospitalario. Estas predicciones se realizarán con datasets generados a partir del histórico de mensajería HL7 del hospital.

Una vez conseguido este dataset y creados los modelos, tal y como hemos comentado debiéramos disponer de un servicio suscriptor que reciba los datos en tiempo real, extraiga los datos más importantes y genere la línea de datos normalizada para alimentar el modelo de forma asíncrona y recibir así la predicción en el punto en el que nos encontremos.

Una secuencia simplificada normal de mensajes de un paciente podría ser la siguiente

1. ADT_A04: Ingreso en urgencia.
2. ADT_A02: Traslado (a punto de atención triaje)
3. ORU_R01: Triaje en urgencia (determinación del nivel de gravedad)
4. ADT_A02: Traslado
5. ADT_A16: Alta médica Urgencia
6. ADT_A03: Alta administrativa urgencia (por ingreso en hospitalización)
7. ADT_A01: Ingreso en hospitalización
8. ADT_A02: Traslado
9. ADT_A02: Traslado
10. ITRI01: Solicitud de intervención
11. ITRI03: Programación de intervención

12. ITRI02: Registro de intervención
13. ADT_A02: Traslado
14. ADT_A16: Alta médica
15. ADT_A03: Alta administrativa

La clave del proyecto para conseguir llegar al objetivo es estudiar el modelo de datos de un paciente que acude a urgencias, ingresa en hospitalización, es intervenido y dado de alta, y observar en qué puntos se dispone de la información necesaria para realizar predicciones. Es decir, se tiene que definir qué variables hay en cada punto, qué dataset se puede generar en base a esas variables y por lo tanto qué modelos se pueden generar.

Por ejemplo, es difícil predecir algo en el primer punto, ya que la información de la que disponemos es poca. En el momento del ingreso disponemos de datos como:

- Edad (39)
- Sexo (M)
- Servicio (Ginecología)
- Sección (Urgencia ginecológica)
- Unidad de enfermería (Urgencia)
- Etc.

En el tercer punto, ya disponemos de un dato muy importante: nivel de gravedad o triaje, que es un valor catalogado de 1 a 5, donde el 1 es el mayor. En este punto, al recibir el nivel de triaje podríamos predecir si va a requerir box, por ejemplo, o cuánto tiempo de box va a necesitar. Para ello debiéramos haber generado un dataset en el que con esos datos relevantes recibidos hasta ese momento nos ayude a predecir, por ejemplo:

- Tiempo en urgencia
- Necesidad de box
- Necesidad de unidades críticas
- Alta a hospitalización o alta a domicilio

El paciente sigue su circuito en el centro y recibe por ejemplo su alta administrativa. En este momento tenemos más datos reales en su dataset normalizado, que enviaremos a otros modelos para saber por ejemplo y con más certeza el tiempo en hospitalización. Aquí dispondremos de datos importantes como el motivo de alta, y otros datos relevantes derivados del circuito que ha hecho el paciente en la urgencia:

- Si ha pasado por box.
- Si ha pasado por unidad crítica.
- Cuál ha sido su tiempo en la urgencia.
- Cuantas veces ha pasado por el proceso de triaje.
- Etc.

En el momento del alta administrativa y en función del tipo de alta:

- Alta a domicilio.
- Alta a hospitalización.

Es decir, se irá completando un dataset incremental en cada punto del proceso, y se irá preguntando a los diferentes modelos creados en cada punto con el objetivo de disponer de ciertas predicciones que puedan ayudar en la gestión del paciente.

Generación de datasets

Los mensajes HL7 hacen referencia a acciones atómicas sobre el episodio de un paciente. Es decir, una apertura de episodio se puede realizar con un ingreso, que genera un mensaje ADT_A01 o ADT_A04 en base a si el episodio es de hospitalización o de urgencia. Existen otras aperturas como (ADT_A05) para ingresos en hospital de día que no tenemos en cuenta.

Es muy importante entender, que la mensajería es secuencial en base a las interacciones en los sistemas de historia clínica y estación médica de la organización y que el mensaje llega en formato HL7 en el payLoad del mensaje (campo que hace referencia al contenido). Por lo tanto, los mensajes deben ser procesados de forma consecutiva, sin ser consecutivos los mensajes de un paciente. Más adelante explicaremos cómo hemos procesado esa información generando grandes estructuras de datos en memoria, generando información que hemos considerado importante, con dos procesadores basados en reglas que nos han

ayudado a obtener información relacionada con el tiempo en unidades, recorrido del paciente, etc. Los datasets generados son:

- **Dataset (A)** que incluye en una línea toda la información relacionada con cada episodio que genera un circuito de paciente. Sobre este dataset se aplican técnicas de aprendizaje supervisado, es decir, en el propio dataset figuran las columnas con las variables a predecir. Como se explica más adelante, el procesador recorre todos los mensajes HL7 para generar una estructura compleja en memoria que son exportadas a formato JSON con cada episodio (urgencia u hospitalización), o se genera un episodio combinado (urgencia + hospitalización) en caso de que sean dependientes. Es importante destacar que se aplican reglas para vincular estos episodios, ya que no existe dato alguno que los vincule. Se ha considerado que una hospitalización está vinculada a una urgencia cuando el alta de urgencia ocurre casi al mismo tiempo que el ingreso en hospitalización y el motivo de alta es ***alta por hospitalización (valor 10).***

  En este dataset incluimos variables extraídas de cada tipo mensaje de la secuencia que podemos englobar en dos tipos:

  - Datos puros: directamente extraídos de los mensajes, durante los pasos de anonimización o pre-procesado:
    - Anonimización: Datos como el episodio (cifrado), CIC (cifrado), edad o sexo, es decir, datos que salen del segmento PID que ha sido eliminado de los mensajes.
    - Pre-procesado: Datos relacionados con los segmentos PV1 e ITR, que vienen directamente informados en los mensajes. Ponemos algunos ejemplos en la Tabla 2:

Tabla 2

Variables relacionadas con los segmentos PV1 e ITR

Variable	Definición
InErDischargeCode	Motivo de alta en urgencia
InHospitalDischargeCode	Motivo de alta en hospitalización
InHosputalNursingUnit	Código de unidad de enfermería de ingreso en hospitalización
InErSurgeryInterventionRegisterProcedureService	Código de procedimiento de intervención quirúrgica en urgencia
InErUniqueBedCode	Código único de ingreso en urgencia
InErTriajeObservationIdentifierCode	Nivel de triaje que adopta los valores: 1. Muy Grave 2. Grave 3. Moderado

	4. Leve 5. Muy leve

- Datos calculados: Después de la realización de un análisis exhaustivo de los datos y de estudiar los diferentes casos de uso existentes en el estado del arte se ha concluido que existen algunos datos que se pueden calcular en base al pre-procesado secuencial de los mensajes. Estos datos no están incluidos en los mensajes, pero sí los valores a partir de los que se calculan. En la Tabla 3, se enumeran lo más importantes y se explica cada dato calculado y el hecho que hace intuir su importancia predictiva:

Tabla 3

Variables más importantes

Variable	Definición
InErNumberOfTriajes	Número de triajes durante el circuito de urgencia. Es un contador de los mensajes tipo ORU que recibe un paciente durante un episodio de urgencia
InErNumberOfTransfers	Número de mensajes de traslado durante su circuito en urgencia. Se calcula este dato ya que el número de traslados puede indicar de alguna manera la gravedad, ya que un paciente no grave de consulta ambulatoria de urgencia contará con un máximo de 2 traslados
InErTotalElapsedHours	Dato que se calcula a través del dos timestamps, el del mensaje de ingreso en urgencia y el del mensaje de alta en urgencia. Convertimos la resta de estos valores a un float que nos indica las horas en urgencia del episodio concreto
InErTimeSlotStart	Se han categorizado 24 horas del día en diferentes slots de tiempo. Entre las 0 am y las 6 am, entre las 6 am y las 12 pm, entre las 12 pm y las 16 pm, entre las 16 pm y las 20 pm y entre las 20 pm y las 0 am. Este valor puede arrojar también cierta urgencia en slots menos frecuentes. Es decir, si se acude a las 4.00am al servicio de urgencia se puede intuir que existe más urgencia que si se acude a las 11.00am (hora punta).
InErTimeSlotEnd	
InHospitalTimeSlotStart	
InHospitalTimeSlotEnd	
InHospitalNumberOfTransfers	Número de traslados en hospitalización. Se calcula este valor de igual forma que calcula el número de traslados en la urgencia y con el mismo objetivo
InHospitalTotalElapsedHou	Número de horas en hospitalización, calculado a partir de

rs	los timestamps de ingreso y alta
InHospital	Valor booleano que indica si ese registro cuenta con episodio de hospitalización
InEr	Valor booleano que indica si ese registro cuenta con episodio de urgencia
InHospitalHasSurgeryIntervetionRequest	Dato booleano que se escribe como true si tiene una solicitud de intervención quirúrgica durante su episodio de hospitalización y urgencia respectivamente
InErHospitalHasSurgeryIntervetionRequest	
InHospitalHasSurgeryIntervetionRegister	Al igual que el anterior, este dato se escribe como true si se recibe un registro de intervención durante el episodio de hospitalización y urgencia respectivamente, es decir, si el paciente es operado
InErHasSurgeryIntervetionRegister	
InHospitalHasSurgeryIntervetionScheduling	Al igual que las anteriores este dato se escribe a true si el paciente recibe durante su episodio una programación de intervención, es decir si se le planifica una operación
InErHasSurgeryIntervetionScheduling	
InHospitalSurgeryInterventionRegisterElapsedMinutes	Tiempo calculado en minutos de la duración del proceso de intervención desde el ingreso hasta su registro de intervención. Al igual que los anteriores uno hace referencia al episodio de urgencia y otro al de hospitalización
InErSurgeryInterventionRegisterElapsedMinutes	
InHospitalTransfers	Se ha considerado este dato como muy importante. Se guarda la cadena de traslados en un campo: 1. Guardando tres datos de cada tralslado (CódigoUnicoOrigen#CodigoUnidoDestino#UnidadDestino) 2. Separando por ';' cada traslado. Es decir, se hace un append cada vez que se recibe un traslado. Esto permite después procesar los datos en Python y obtener datos diferentes registros con el destino del paciente (para este trabajo se tendrán en cuenta sólo las unidades de enfermería) y se generarán columnas de **Origen, Destino y Trayecto** acumulado. Se realiza el mismo cálculo durante el episodio de urgencia, pero en este caso no aporta tanto valor, ya que la unidad de enfermería

	para todos los códigos únicos de urgencia es la misma
InHospitalElapsedHoursInCritics	Es la suma de las horas en unidades críticas del episodio de hospitalización calculada a través de los timestamps de ingreso, traslados y altas. Esta variable puede resultar una buena variable a predecir ya que indica la ocupación por parte de un paciente de recursos críticos
InHospitalElapsedHoursNonCritics	Es la suma de las horas en unidades no críticas del episodio de hospitalización calculada a través de los timestamps de ingreso, traslados y altas
InHospitalNursingUnit9001ElapsedHours	Cálculo por cada paciente del tiempo por cada unidad de enfermería por la que pasa. El 9001 hace referencia al código de unidad, por lo que se dispone de un campo por cada código y se calcula con el timestamp de ingreso o traslado a esa unidad como destino y el traslado como origen de esa unidad o el alta

- **Dataset (B)** que se generará de igual manera, pero añadiendo una línea adicional con cada traslado, ver Tabla 4. Esto lo haremos con el objetivo de poder realizar predicciones con los mensajes de traslado (**ADT_A02**), ya que se dispone de datos relacionados con los tiempos en cada unidad (que se calcula en base a los timeStamps de los mensajes), y se conocen además el origen y el destino. Esta técnica permite también obtener más filas en nuestro dataset, por lo que, aunque su propósito predictivo es diferente se puede considerar como técnica de over-sampling. Esta técnica permitirá predecir el código de traslado de la siguiente unidad, a partir de todas las variables predictoras, además del recorrido acumulado, el origen, el destino y los tiempos acumulados en unidades.

Tabla 4

Datos de ejemplo de registros por traslado

	CIC	Origen	Destino	Recorrido
1	X	9150	9250	9150;9250
2	X	9250	9401	9150;9250;9401
3	X	9401	9500	9150;9250;9401;9500

Como se puede observar, se obtiene un nuevo registro con cada traslado que además genera tiempos en unidades acumulados que servirán para predecir el próximo destino de un paciente.

3.2. Objetivos específicos

Con la intención de alcanzar el objetivo general de este trabajo, se detallan los siguientes objetivos específicos:

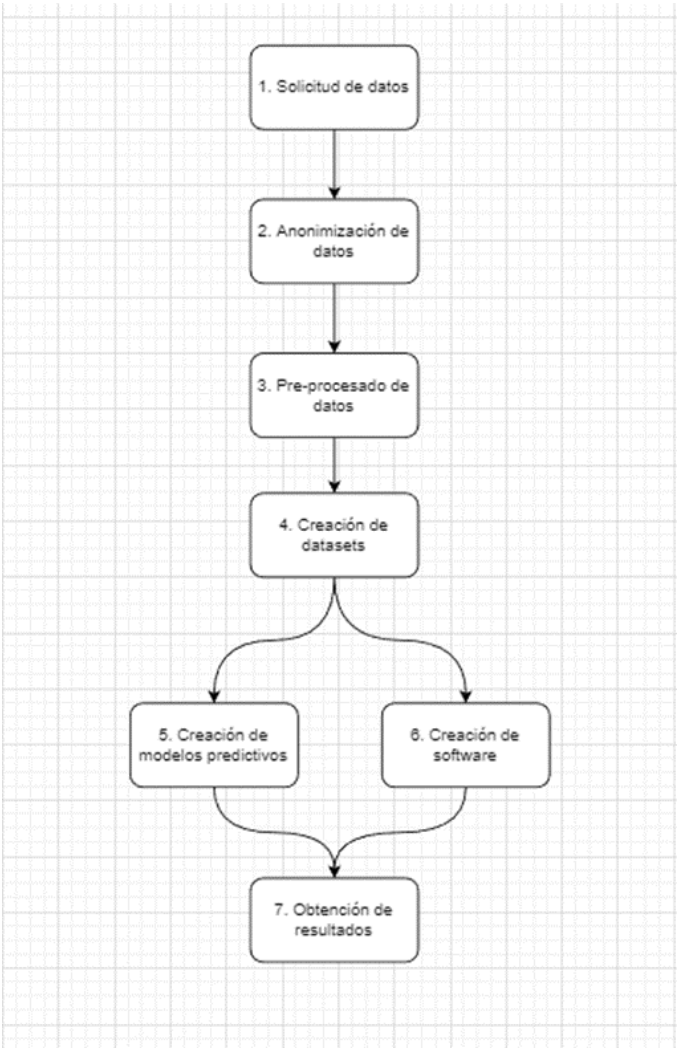
- Extraer información de valor de los mensajes HL7 para poder realizar modelos predictivos.
- Implementar un modelo de trabajo para llevar a cabo las siguientes predicciones a partir de los datos extraídos directamente de la mensajería HL7:
    - Predecir la mortalidad (alta por muerte) de un paciente a partir del **dataset (A),** que define los episodios hospitalarios.
    - Predecir con los datos proporcionados por el **dataset (B)** si el próximo paso que va a tener el paciente en el circuito de hospitalización es un alta (alta por muerte, alta por ingreso urgencias, alta del hospital, etc.)
- Analizar los diferentes resultados obtenidos por los algoritmos empleados en las predicciones.
- Comparar la precisión e idoneidad de los diferentes métodos o algoritmos realizados.

3.3. Metodología del trabajo

La metodología de trabajo se compone de las siguientes etapas (ver Figura 11):

Figura 11

Esquema de la metodología de trabajo



1. ***Solicitud de datos:*** Ha sido solicitado el histórico de mensajería HL7 del último año. Para ello la subdirección de sistemas de la OSI Ezkerraldea Enkarterri Cruces ha exigido:

   - Cifrado de los datos identificativos necesarios (CIC)
   - Anonimización de los mensajes.
   - Firma de compromiso de confidencialidad por parte de los participantes en el proyecto.

2. ***Anonimización de datos***: En esta parte del proceso se ha creado un software específico con tecnologías *.NET* que realiza las siguientes funciones:

   a. Conecta con el repositorio de mensajes

   b. Recorre el repositorio realizando estas tareas:

i. Cifra los datos 'episodio' y 'cic' del segmento *PID* almacenando el resultado en memoria

ii. Extrae los datos de edad y sexo del segmento *PID* completo almacenándolo en memoria

iii. Elimina el segmento *PID* completo

iv. Persiste el mensaje sin segmento *PID* con el resto de datos extraídos en una nueva tabla

La OSI ha proporcionado mensajes del entorno pre-producción con datos no pertenecientes a pacientes reales para realizar esta tarea. Una vez implementado y testeado para cada tipo de mensaje, se ha entregado el software y se ha preparado el entorno (base de datos destino) para que la herramienta funcione de forma correcta. La OSI ha procedido a anonimizar los mensajes y ha validado la correcta anonmización antes de ser entregados. El proceso de anonimizado supera las 6h en tiempo de ejecución.

3. ***Pre-procesado de datos:*** Se ha desarrollado un software también basado en tecnologías *.NET* que mapea en una estructura de clases todos los mensajes:

   a. Se ha trabajado con *.NET* para poder trabajar con *LINQ*[9] de cara a pre-filtrar y elegir los pacientes candidatos, descartando aquellos que no cumplen con episodios de urgencia o de hospitalización.

   b. Agrupa los mensajes de los episodios de urgencia y hospitalización que estén correlacionados.

   c. Genera la clase que va a alimentar a los datasets partiendo de los datos del punto anterior y contiene toda la información necesaria del circuito realizado por el paciente (desde el ingreso hasta el alta).

4. ***Creación de datasets:*** La herramienta anterior permite guardar los datos filtrados creando registros únicos por cada combinación de episodios de urgencia y hospitalización de un paciente. También se ha implementado la posibilidad de generar registros adicionales en aquellos casos que existan traslados en hospitalización.

5. Creación de Modelos Predictivos: Antes de la realización de los modelos predictivos ha sido necesario efectuar un tratamiento de nulos y variables categóricas sobre las tablas creadas. A partir de ahí, se comienza a realizar las predicciones previamente enunciadas, las cuales son:

   a. Predecir la mortalidad de un paciente a partir de los episodios recogidos en el dataset previamente creado (dataset A).

[9] LINQ o Language Integrated Query son un conjunto herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: objetos, xmls, bases de datos, etc…

b. Prever el alta de un paciente en base a los datos obtenidos en el conjunto de traslados de los usuarios ingresados en el hospital. (dataset B)

Para efectuar estas predicciones se han utilizado los siguientes modelos de clasificación:

i. Naive Bayes.

ii. Árboles de decisión.

iii. Random forests.

iv. Gradient Boosting Decision Tree (GBDT)

v. Red neuronal perceptrón multicapa (MLP)

6. ***Obtención de resultados:*** Partiendo de las predicciones anteriormente desarrolladas, se obtienen las diferentes métricas sobre los rendimientos de los modelos; realizándose un análisis comparativo de estos valores, seleccionando el algoritmo que mejor encaja para cada predicción.

7. ***Creación del software:*** Debido a que tanto la generación de los datasets como de los modelos predictivos han supuesto más tiempo de trabajo que el inicialmente planteado, este último punto del proceso no se ha podido realizar debido, en parte, a falta de recursos y tiempo. En las previsiones iniciales se incluía este apartado; el cual consiste en el diseño de un software que sea capaz de leer un mensaje HL7 en tiempo real y llevar a cabo las transformaciones necesarias sobre dicho dato y la posterior predicción.

4. Identificación de requisitos

En este apartado se detalla la metodología a seguir para conseguir un dataset normalizado, agrupado por pacientes, a partir de un histórico de mensajería HL7 anonimizado. Los datasets se crean a partir de un histórico secuencial de mensajería HL7, donde es necesario entender cómo es el paso (circuito) de los pacientes por un hospital, que dada la mensajería de la que se dispone se agrupa en pacientes de Urgencia y Hospitalización.

Del mismo modo, se explican los pasos efectuados para poder realizar las predicciones, así como las diferentes técnicas de evaluación empleadas para observar el comportamiento de los modelos predictivos.

4.1 Histórico de mensajería

Como hemos comentado, el hospital dispone de un histórico de la mensajería relacionada con hospitalización (camas y bloque quirúrgico) y urgencia. Disponemos de los siguientes tipos de mensaje:

- ADT_A01. Ingreso en hospitalización.
- ADT_A04. Ingreso en urgencia.
- ADT_A06. Ingreso en hospital de día.
- ADT_A03. Alta administrativa.
- ADT_A16. Alta médica.
- ADT_A02. Traslado.
- ADT_A31. Modificación de ingreso.
- ADT_A11. Anulación de ingreso.
- ADT_A12. Anulación de traslado.
- ADT_A13. Anulación de alta.
- ADT_A14. Creación de solicitud de ingreso.
- ADT_A25. Anulación de alta médica.
- ITR_I01. Solicitud de intervención quirúrgica.
- ITR_I02. Registro de intervención quirúrgica.
- ITR_I03. Programación de intervención quirúrgica.
- ORU_R01: Triaje en urgencia.

Existen otros mensajes que no tendremos en cuenta, ya que su uso es residual y desestimaremos los registros en los que aparecen: traslados al exterior, traslados desde el exterior, etc.

Salvo excepciones, cada tipo de mensaje dispone de unos segmentos generales, y unos específicos que aportan los datos de la operación correspondiente. Así, por ejemplo, un ingreso dispone de información relativa al paciente (PID) e información relativa al episodio (PV), en cambio una solicitud de intervención dispone de su segmento PID, su segmento PV, además del segmento SIN, que proporciona datos relativos al diagnóstico y el tipo de intervención y recursos necesarios.

Es muy importante entender, que los mensajes son generados en un momento concreto del tiempo y que es estrictamente necesario recibirlos, almacenarlos y procesarlos en el orden de creación. Esto se justifica con el siguiente ejemplo: si queremos extraer el tiempo de alta

en el momento de hospitalización y recibimos primero el alta y después el ingreso, no disponemos del timestamp de ingreso.

En el caso de extracción de datos para la posterior creación de datasets no debiera ser estrictamente necesario ya que el tiempo de operación va implícito en el propio mensaje, y lo utilizaremos para rellenar datos de la estructura generada, pero en una aplicación en tiempo real en la que se realicen predicciones si lo sería.

4.2 Circuitos del paciente.

A continuación, se muestran los posibles circuitos realizados por un paciente tanto por urgencia como por hospitalización.

4.2.1 Paciente de urgencia

En la Figura 12 se muestra el circuito de un paciente en la urgencia y cómo se realizarán las capturas de datos y las predicciones en base a los mensajes. Se utilizan los códigos:

- E: Evento o mensaje HL7.
- C: Proceso de captura de datos y normalización
- M: Modelo predictivo
- R: Proceso de captura y almacenamiento del resultado del modelo.

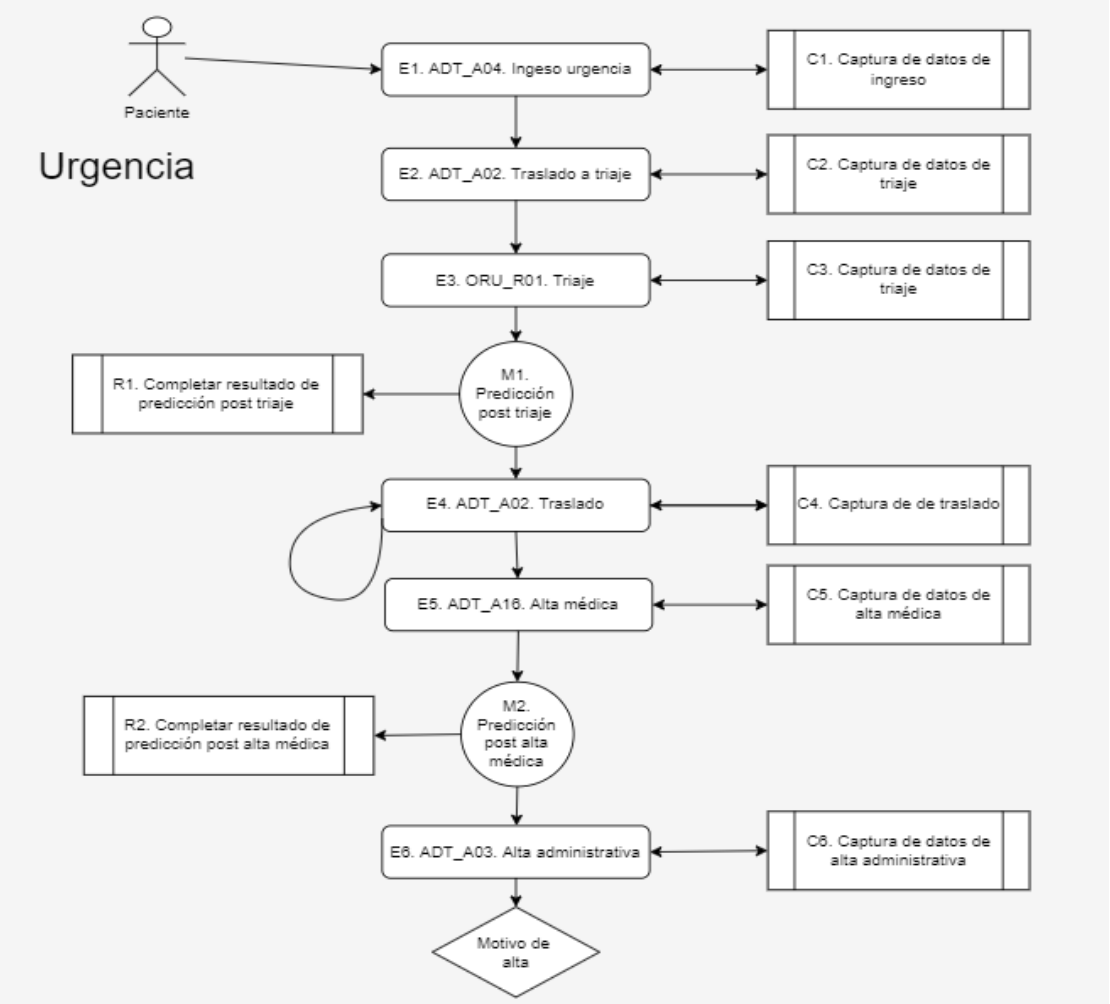
El funcionamiento es el explicado en el apartado anterior: El paciente ingresa (E1) y se comienzan a capturar datos del ingreso (C1), es trasladado a triaje (E2) y se capturan más datos relacionados con el traslado (C2). Esto se debe a que este traslado nos puede aportar información relevante, y es que existen diferentes puntos de triaje:

- TRGIN. Triaje ginecológico y obstetricia
- PU1: Pendiente de ubicar 1. Pacientes de la urgencia general.
- PU2: Pendiente de ubicar 2. Durante la pandemia se ha utilizado este punto para realizar el triaje de los pacientes con síntomas COVID.
- TRPED: Triaje de pediatría.
- BOXES: Si el paciente no ingresa en uno de los anteriores, puede ingresar directamente en un BOX, hecho que indica que existe mayor gravedad, ya que no ingresa como paciente reconocible en consulta ambulatoria de urgencia.

Una vez triado (E3), recibimos el nivel de gravedad, y aunque este diagrama es una aproximación, en este punto podríamos disponer de variables predictoras suficientes para alimentar al primer modelo entrenado (M1) y recibir la primera predicción.

Figura 12

Circuito de urgencia



El paciente es trasladado en una o diferentes ocasiones a través de la urgencia. El primer traslado pude ser:

- A una consulta ambulatoria, si su nivel de gravedad (triaje) está en el rango [3,5].
- A un box, si su nivel de gravedad está en el rango [1,3].

Se capturarán los traslados y se extraerán los datos de origen y destino para almacenar el tiempo en cada punto de atención en boxes o consultas ambultatorias.

El circuito del paciente de urgencia se completa con el alta adminisitrativa (E6), pero previamente ha recibido un alta médica (E5), que aportará datos para alimentar también al siguiente modelo. Aquí capturaremos por ejemplo el motivo de alta entre los que destacamos tres:

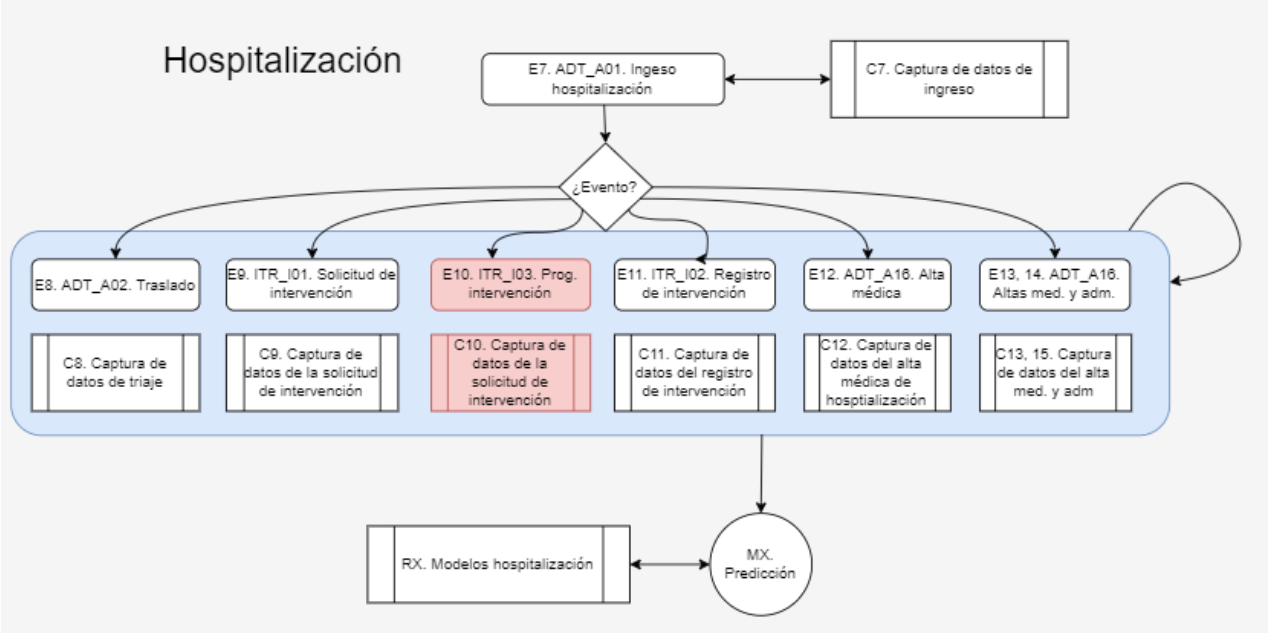
- Alta a domicilio.
- Alta a hospitalización: El paciente recibirá posteriormente un ingreso a hospitalización. Vincularemos el episodio de urgencia con el de hospitalización para poder hacer un ***merge*** de los dos episodios. Cabe destacar que cada episodio es independiente, y no hay manera de vincular su estancia en el hospital (urgencia + hospitalización) de otra forma.
- Alta por éxitus (defunción).

4.2.2 Circuito de hospitalización.

El circuito de hospitalización (ver Figura 13) ofrece muchos más datos que el de la urgencia, pero la captura de los datos es más compleja, ya que no es una secuencia tan estructurada.

Figura 13

Circuito de hospitalización



Los pacientes de hospitalización ingresan habiendo o no pasado por la urgencia, pero por norma general un paciente hospitalizado ingresa por dos motivos:

- Ha pasado por la urgencia y ha recibido un alta a hospitalización
- Tiene programada una intervención quirúrgica, que requiere de hospitalización.

En ambos casos se genera un episodio de hospitalización con su ingreso, y lo que haremos será capturar los diferentes tipos de eventos que nos ofrezcan información relevante para en el procesado posterior crear los datasets. Estos eventos son los siguientes:

- E8 (ADT_A02 - Traslado): Capturaremos los traslados para conocer los puntos de destino de ese paciente. Estos puntos de destino pertenecen a unidades de enfermería que catalogaremos y aunque en sí no ofrezcan datos clínicos, nos pueden revelar información de la gravedad o estancia del paciente. Y es que no es igual de relevante que un paciente pase por la unidad de grandes quemados por ejemplo que por hospitalización en digestivo. Respecto a la captura de estos datos, existen diferentes formas y estructuras para almacenarlos. Estudiaremos si incluir en la tabla de episodios las ubicaciones más relevantes, o si por contrario, generar trazabilidad de cada traslado con una entidad auxiliar para controlar el tiempo en cada unidad. Esta segunda opción ofrece una información más completa pero también cierta complejidad a la hora de procesar los datos para generar los datasets.

- E9 (ITR_I01 – Solicitud de intervención): Es un evento muy importante para nuestra plataforma, ya que en la solicitud de intervención se incluye información clínica y diagnóstica importante como por ejemplo el código CIE. Este código, pertenece a un catálogo estandarizado de diagnósticos en el ámbito salud. Además, existen otros muchos datos importantes como algunos relacionados con los recursos necesarios para la intervención quirúrgica, etc.

- E10 (ITR_I03 – Programación de intervención): La programación de intervención es un evento que no ofrece mucha información, salvo la fecha de programación. Estudiaremos si lo utilizamos, ya que es posible que podamos extraer información relacionada con la urgencia de esta, si catalogamos por ejemplo intervenciones en rangos de tiempo a:

    - Una semana de la solicitud (Urgencia alta)
    - Entre una semana y un mes (Urgencia media alta)
    - Más de un mes y menos de tres (Urgencia media baja)

- Más de tres meses (Urgencia baja)

- E11 (ITR_I02 – Registro de intervención): Extraeremos información sobre el resultado de la intervención. Puede ser un buen punto para predecir la necesidad de hospitalización con los registros de nuevos pacientes alimentando al sistema.

- E12 (ADT_A16). Al igual que en urgencia, refleja el alta médica. Extraeremos datos de alta médica y puede ofrecer también un punto de predicción para predecir próximas hospitalizaciones o episodios de urgencia si controlamos los pacientes que han reingresado después de este evento también en rangos temporales:

    - Una semana del alta

    - Entre una semana y un mes

    - Más de un mes y menos de tres

    - Entre tres meses y un año

- E13 (ADT_A03). Al igual administrativa. En principio los datos que nos ofrece son datos administrativos sumados a los datos de alta médica. Estudiaremos los valores que proporciona para ver si podemos completar el circuito.

4.2.3 Pacientes candidatos

Como hemos comentado, el circuito normal de un paciente tiene un paso urgencia y un paso por hospitalización. Existen otros circuitos como el ingreso en hospitalización para la realización de una intervención programada que nos aporta menos información al sistema. Otro posible circuito es sólo el de urgencia con alta a domicilio que aporta aún menos información.

Además, se parte de un histórico con un corte temporal sin conocer el estado anterior del hospital, por lo que es necesario eliminar a la mensajería de pacientes ya ingresados antes del primer mensaje.

El proyecto se centra en pacientes que han pasado por urgencia y hospitalización ya que podemos vincular ambos episodios a través de los eventos “Alta Urgencia” con motivo de hospitalización y posteriormente “Ingreso en hospitalización”. De esta forma obtendremos un

dataset más completo que nos ayude a generar algunas predicciones en el momento de hospitalización.

4.3 Generación de datasets.

Para la generación del dataset se parte de un histórico de 2.463.363 mensajes HL7 comprendidos entre 26/03/2021 y 10/04/2022. Como se ha comentado anteriormente el objetivo es tener la información del circuito de un paciente (desde que ingresa, urgencias u hospitalización, hasta que obtiene el alta).

Cada mensaje está estructurado en segmentos.

- Encabezado del mensaje
- Tipo de evento
- Identificación del paciente
- Visita del paciente
- Visita del paciente - Información adicional
- Información sobre el diagnóstico
- Alergias

A su vez los mensajes se pueden categorizar en tres grandes grupos:

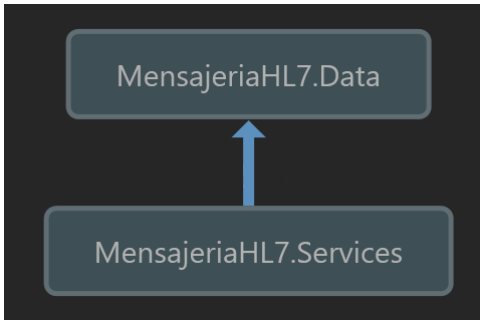
- Mensajes ADT: Administrativos con información sobre ingresos, traslados, altas.
- Mensajes ORU: Mensajes relacionados con el nivel de gravedad del paciente en la urgencia.
- Mensajes ITR: Mensajes relacionados con el bloque quirúrgico. Este tipo de mensajes contienen a su vez segmentos por cada tipo de intervención
    - Segmento de solicitud de intervención
    - Segmento de registro de intervención
    - Segmento de programar/desprogramar intervención
    - Segmento de derivar intervención
    - Segmento de cambio de quirófano de intervenciones

4.3.1 Creación de clases para la traducción de los mensajes

Se ha optado por emplear programación por capas para la realización del traductor de mensajes. En este caso se han empleado dos capas: la capa de datos y la capa de servicios, tal y como se puede apreciar en la Figura 14.

Figura 14

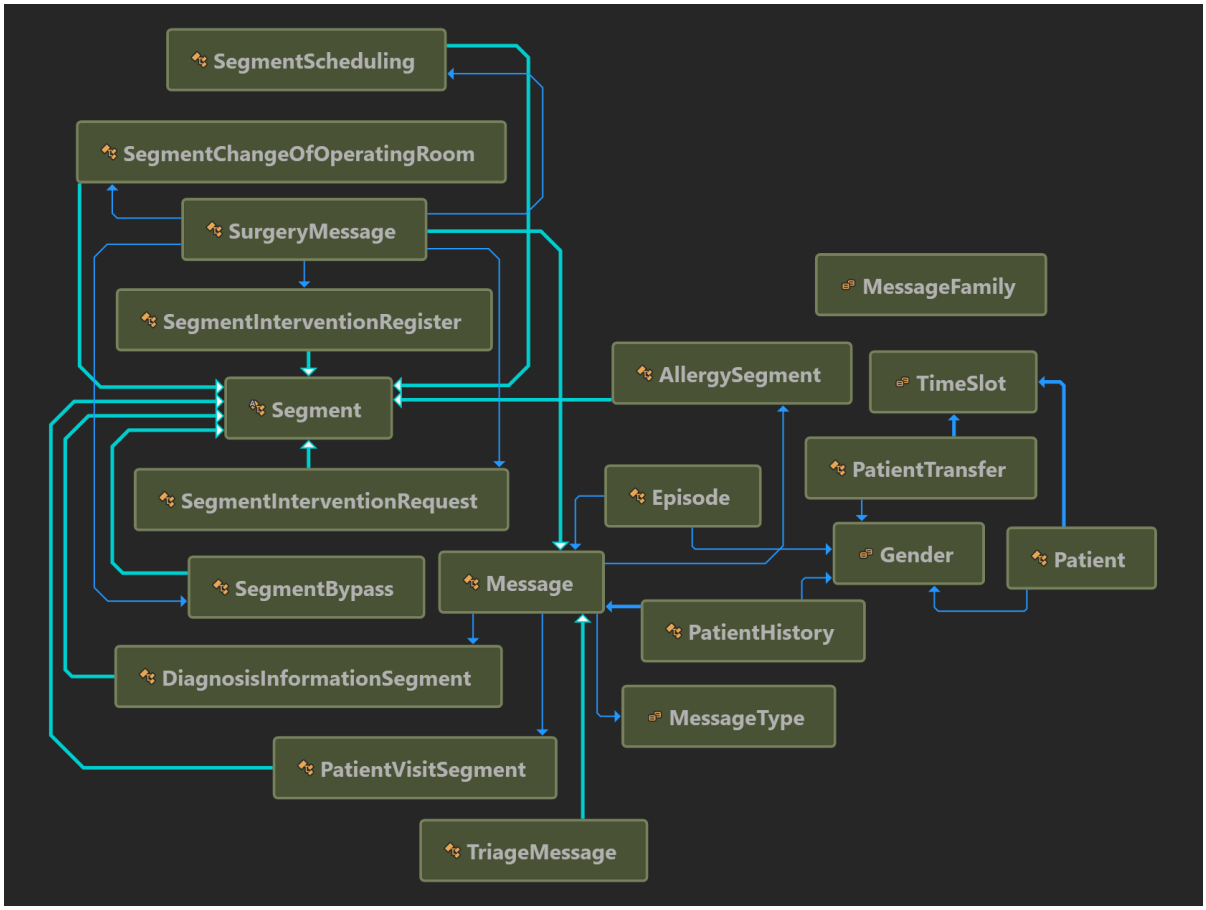
Capa de datos y capa de servicios



La capa de datos se compone de todas las clases necesarias para modelar los mensajes, en la Figura 15 se muestra un diagrama de las mismas.

Figura 15

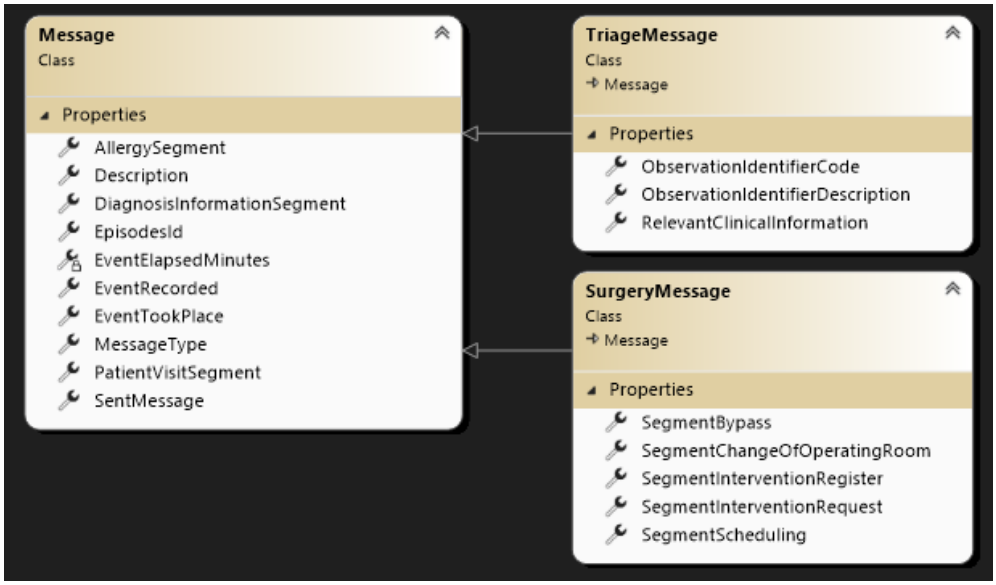
Modelado de clases de la capa de datos



El detalle de la clase mensaje y de las clases que heredan de éste se muestra en la Figura 16.

Figura 16

Modelado de clases para mensajes



Los mensajes se componen de segmentos de información y el detalle de los mismos se muestra en la Figura 17 y en la Figura 18.

Figura 17

Modelado de segmentos para los mensajes del tipo intervención

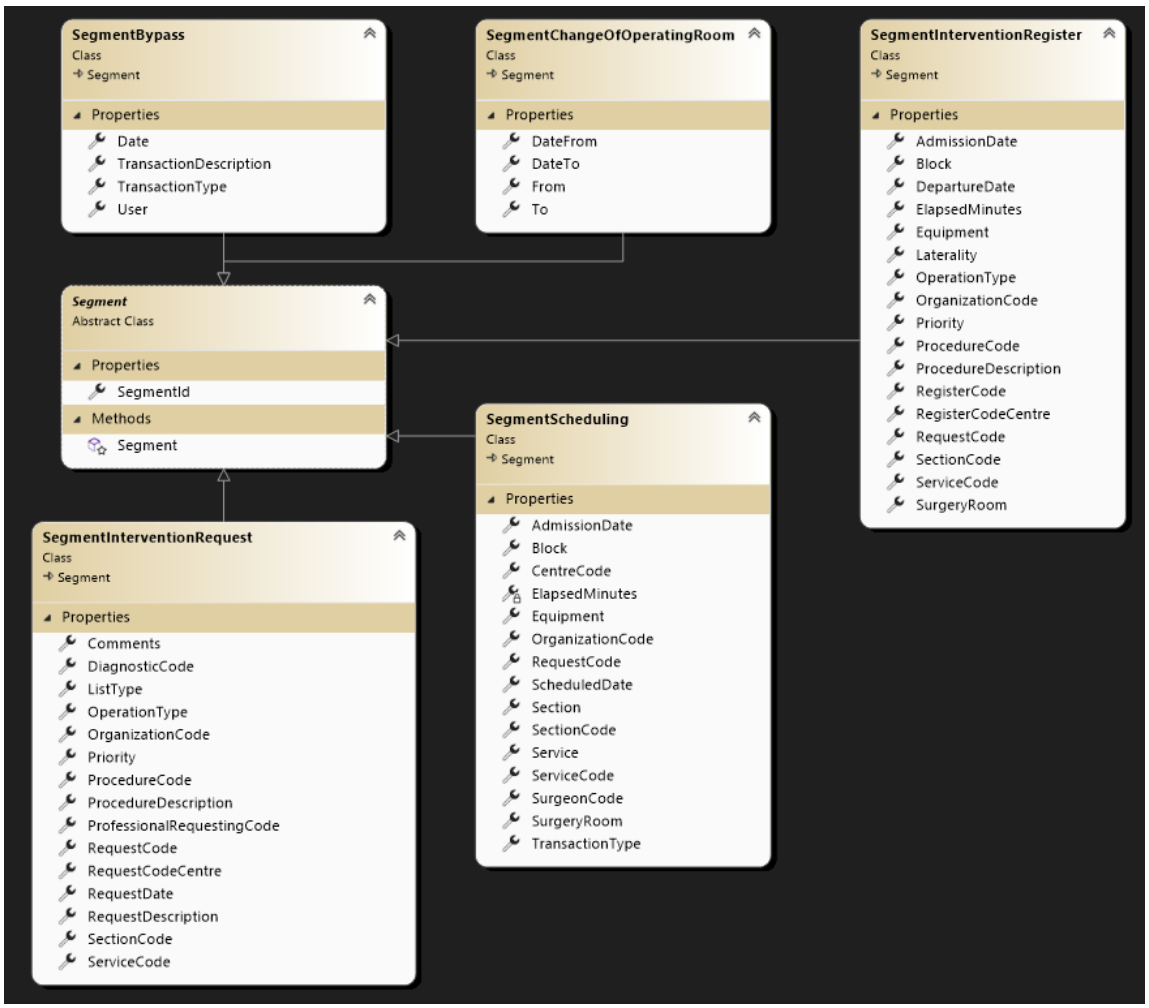


Figura 18

Modelado de segmentos para los mensajes en general

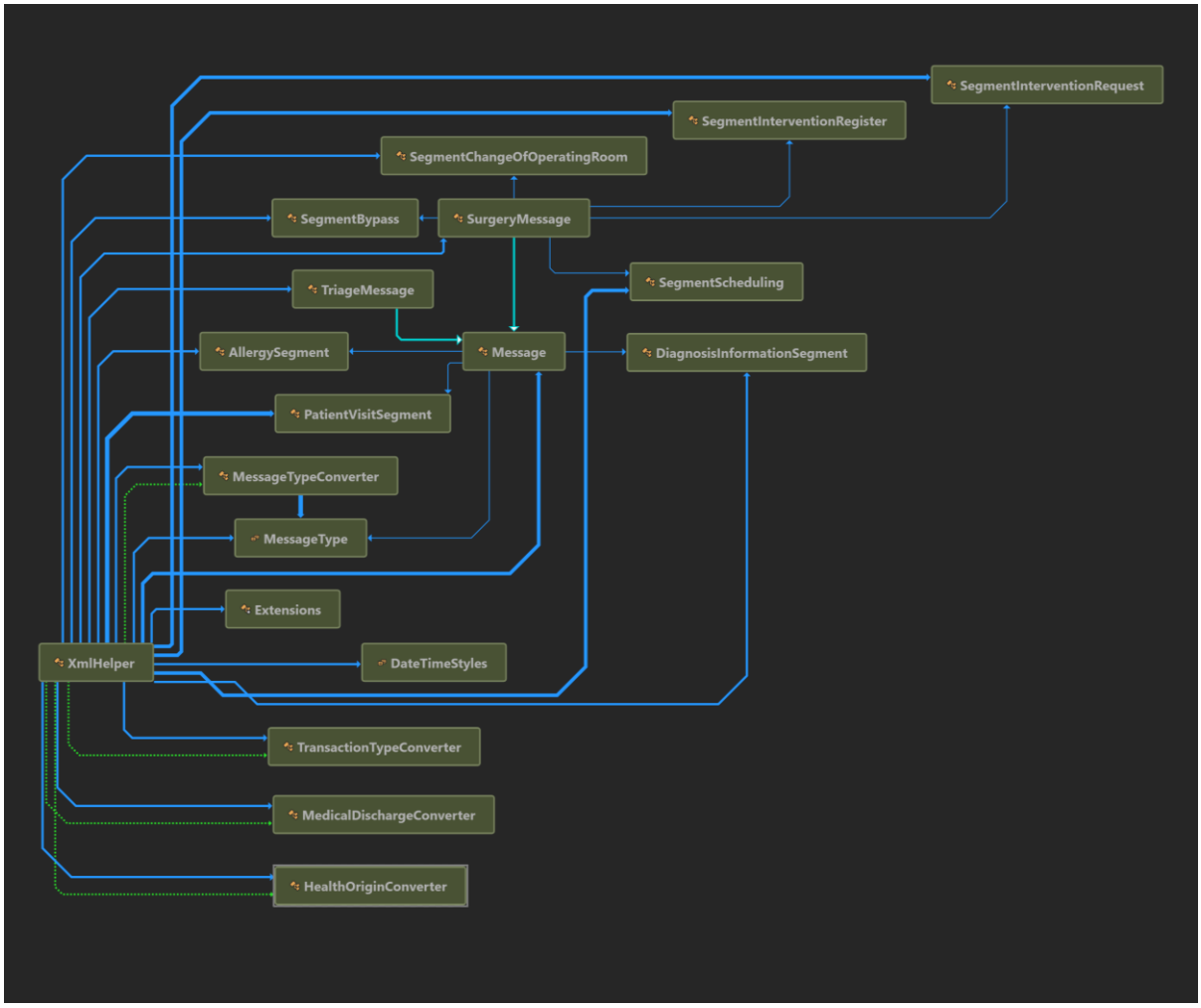
La capa de servicios contiene las clases necesarias para la traducción de los mensajes y la generación de los circuitos de cada paciente. En la Figura 19 se muestra un diagrama de dependencia de la clase XmlHelper que se emplea para la traducción de los mensajes que están en XML[10]. Captura los datos de los mensajes en base a los segmentos que lo componen y almacena la estructura de los mensajes en memoria, con el objetivo de poder

[10] XML es un lenguaje de marcado similar al HTML. Las siglas significan eXtensible Markup Language (Lenguaje de Marcado Extensible) y se trata de una especificación del W3C como lenguaje de marcado de propósito general.

recorrer esta estructura para extraer los datos necesarios para generar las variables de nuestro dataset.

Figura 19

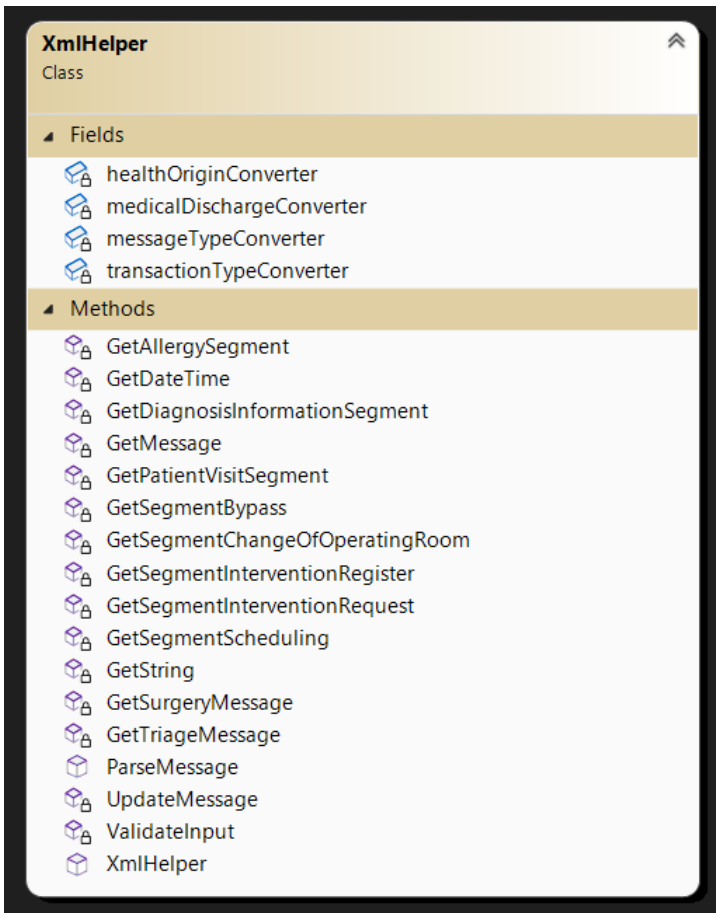
Modelado de las clases necesarias para la traducción de los mensajes



El detalle de cómo está modelada la clase XmlHelper se puede ver en la Figura 20.

Figura 20

Modelado de la clase que lee los mensajes

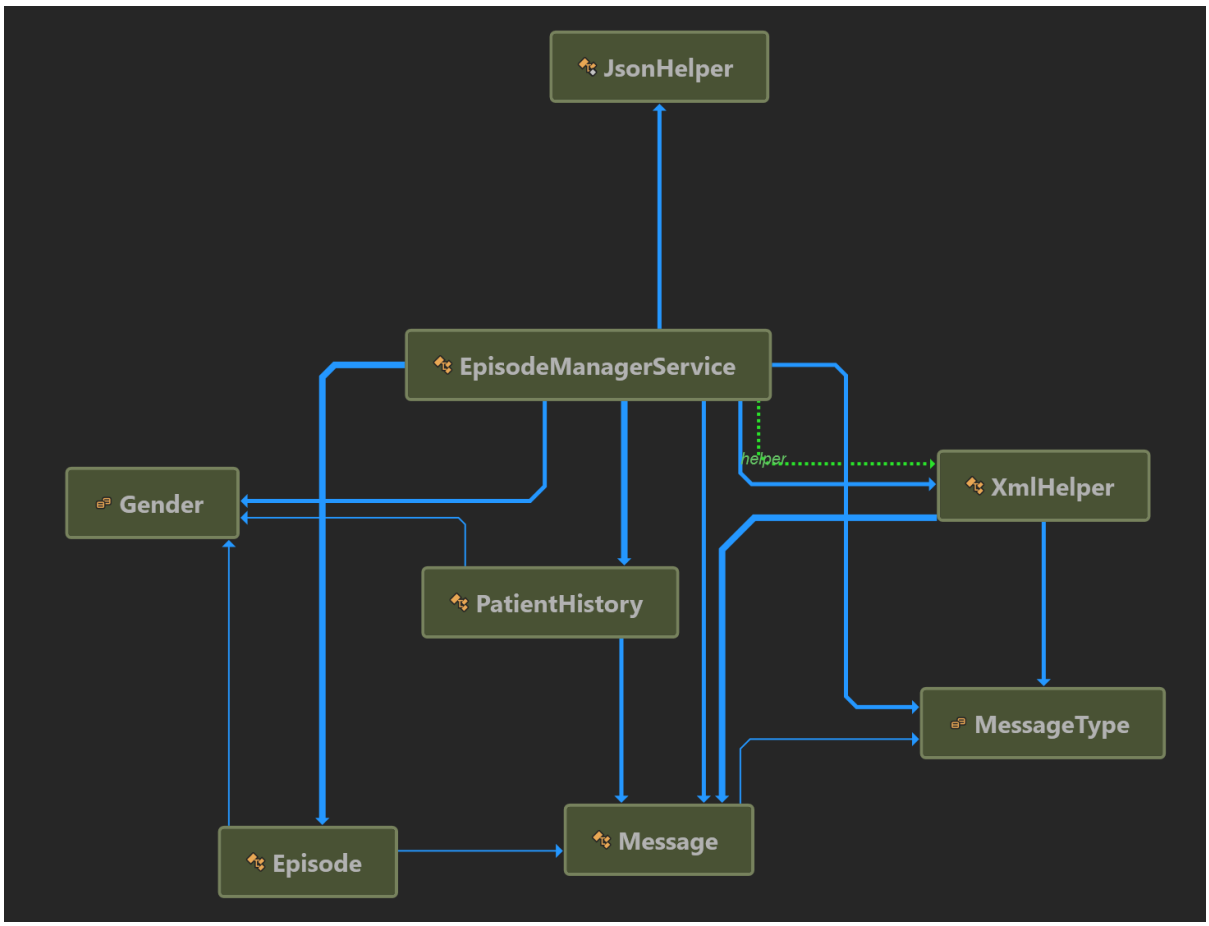


Una vez que se tienen los mensajes traducidos, el siguiente paso ha sido agrupar los mensajes por CIC[11], para ello se ha creado una nueva clase EpisodeManagerService (ver Figura 21) que modela el historial del paciente (ver Figura 21). El número de histórico de pacientes obtenidos es 136.805. El traductor se ha desarrollado con tecnologías .NET. Esto posibilita el uso de LINQ para agrupar y reordenar facilitando así el procesado posterior de los mensajes.

[11] Cada paciente se identifica a través de un número correlativo y no reutilizable: CIC (Código de Identificación Corporativo), lo que permite obtener a través de una única consulta información acerca de todos los episodios, informes y resultados del paciente, independientemente de la modalidad asistencial que lo produjo (hospitalización,, consultas externas, urgencias), y del centro de la red que lo haya tratado (centros de atención primaria, asistencia en las unidades de emergencias, hospitalización a domicilio, etc..)

Figura 21

Diagrama de dependencias de la clase EpisodeManagerService



El detalle de cómo está modelada la clase EpisodeManagerService se puede ver en la Figura 22.

Figura 22

Modelado de la clase EpisodeManagerService

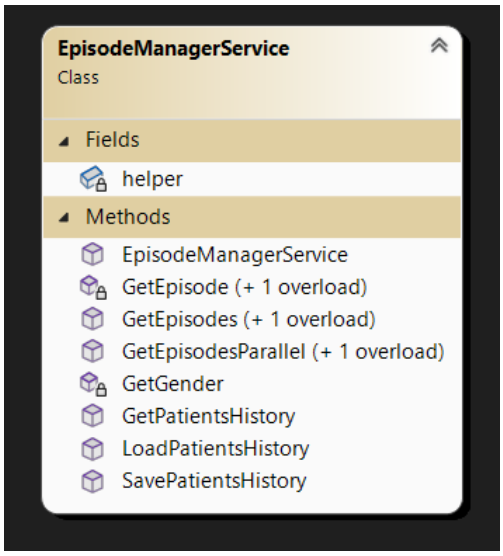
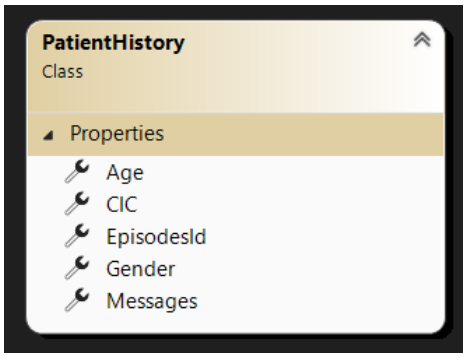


Figura 23

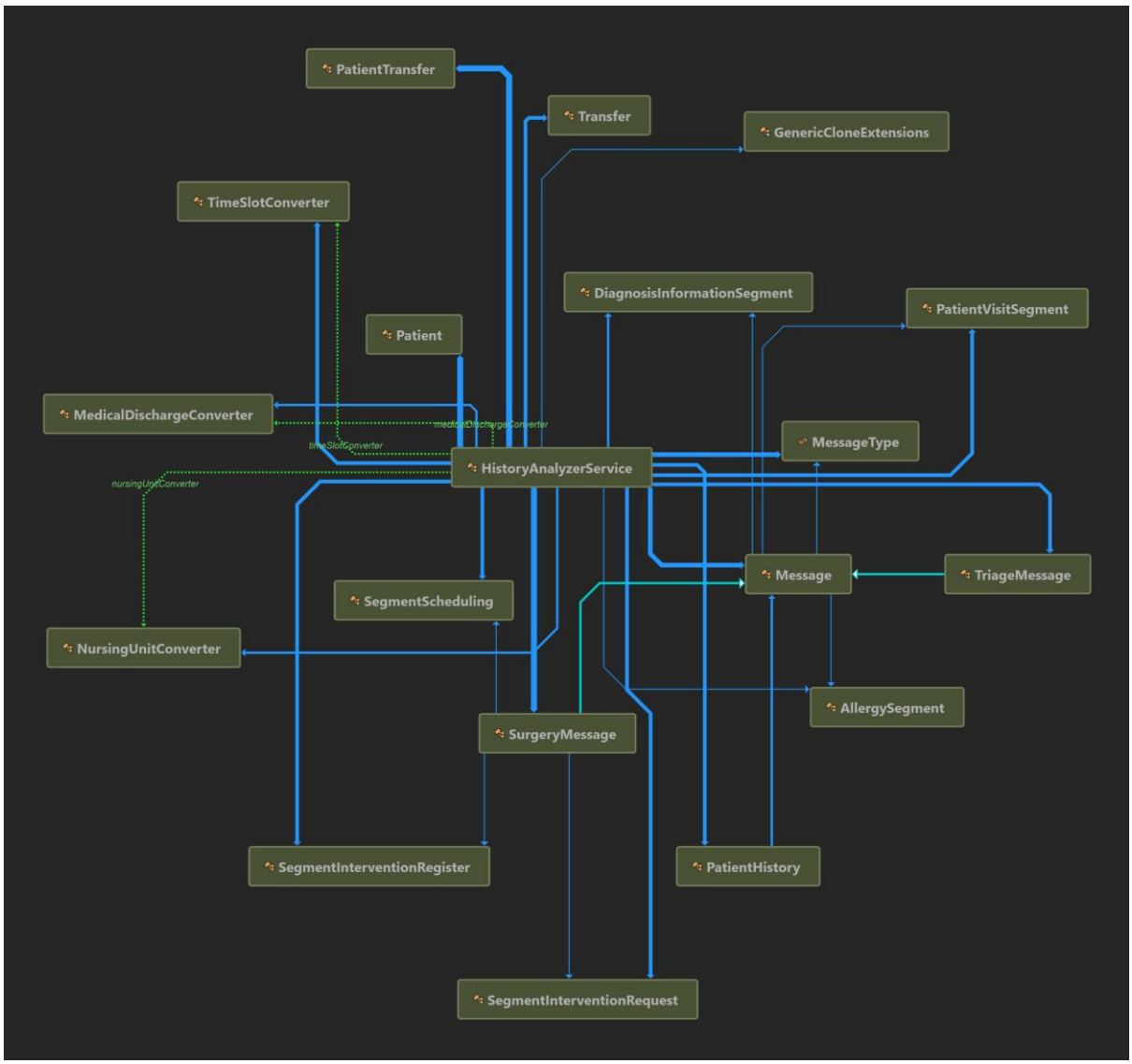
Modelado de la clase que representa la historia del paciente



El tercer paso ha sido “modelar” el circuito del paciente a partir de los datos de la clase anterior, para ello se hace uso de la clase HistoryAnalyzerService (ver Figura 24).

Figura 24

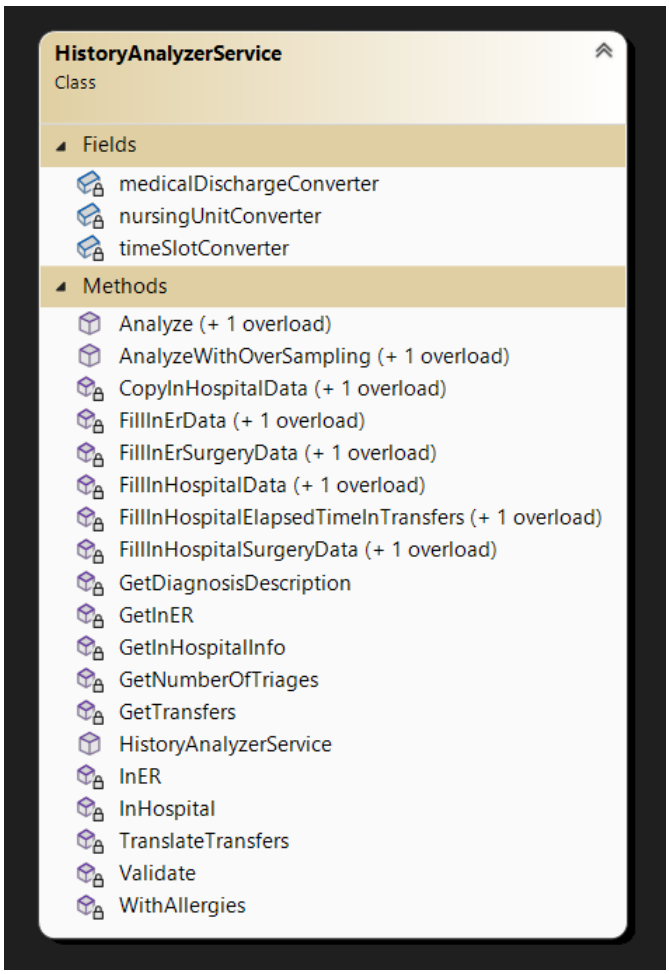
Diagrama de dependencias de la clase HistoryAnalyzerService



El detalle de cómo está modelada la clase HistoryAnalyzerService se puede ver en la Figura 25.

Figura 25

Modelado de la clase HistoryAnalyzerService

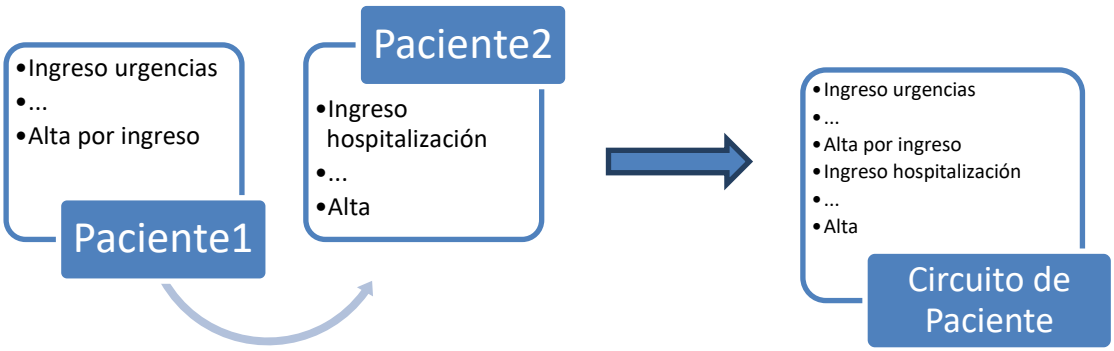


A continuación, se detallan las decisiones tomadas para poder generar dicha clase:

- La primera decisión que se ha tomado ha sido la de filtrar a los pacientes, se han eliminado aquellos que no tengan al menos un ingreso en urgencias y un ingreso en hospitalización.
- La segunda decisión ha sido subdividir la lista de mensajes, cada vez que se detecte un ingreso, ya sea por urgencia o por hospitalización. De esta forma se puede considerar como nuevo paciente al grupo de mensajes comprendidos entre un ingreso y un alta.
- La tercera decisión consistió en unir aquellos pacientes cuya alta era del tipo 10 (alta por ingreso) y el ingreso fuera de hospitalización (ver Figura 26).

Figura 26

Esquema explicativo para unir dos pacientes



Al llevar a cabo estas decisiones se pasan de los 136.085 históricos a 56.841 circuitos de pacientes.

Por cada circuito de paciente (ver Tabla 5) se dispone de la siguiente información: edad, sexo, si tiene alguna alergia, tiempo total en urgencias, tiempo total en hospitalización, tiempo en cada unidad de enfermería, información de traslados, información de triajes, franja horaria del ingreso y del alta, el diagnóstico, información por cada mensaje de intervención, tiempo total en unidades de enfermería correspondiente a críticos, tiempo total en unidades de enfermería que no sean críticos. En total son 172 campos.

Tabla 5

Definición de la clase circuito

PROPIEDADES	DEFINICION
CIC	Cic de paciente
Age	Edad
Gender	Género
WithAllergies	Si tiene alegrias
InErNumberOfTriages	Nº de triajes en urgencias
InErNumberOfTransfers	Nº de traspasos en urgencias
InErTotalElapsedHours	Tiempo total en horas en urgencias
InErTimeSlotStart	Franja horaria de ingreso en urgencias
InErTimeSlotEnd	Franja horaria de alta en urgencias
InErDiagnosisDescription	Descripción del diagnóstico en triaje en urgencias
InErTriageRelevantClinicalInformation	Observaciones en triaje en urgencias
InErTriageObservationIdentifierCode	Código único de cita/solicitud en triaje en urgencias
InErTriageObservationIdentifierDescription	Descripción del código único de cita/solicitud en triaje en urgencias
InHospitalNumberOfTriages	Nº de triajes en hospitalización
InHospitalNumberOfTransfers	Nº de traspasos en hospitalización
InHospitalTotalElapsedHours	Tiempo total en horas en hospitalización

InHospitalTimeSlotStart	Franja horaria de ingreso en hospitalización
InHospitalTimeSlotEnd	Franja horaria de alta en hospitalización
InHospitalDiagnosisDescription	Descripción del diagnóstico en triaje en hospitalización
InHospitalTriageRelevantClinicalInformation	Observaciones en triaje en hospitalización
InHospitalTriageObservationIdentifierCode	Código único de cita/solicitud en triaje en hospitalización
InHospitalTriageObservationIdentifierDescription	Descripción del código único de cita/solicitud en triaje en hospitalización
InHospital	¿Ha sido hospitalizado?
InHospitalNursingUnit	Código de unidad de enfermería en hospitalización
InHospitalNursingUnitIsCritics	Es de críticos el código de unidad de enfermería en hospitalización
InHospitalLocationIdentifier	Descripción corta unidad de enfermería en hospitalización
InEr	¿Ha ido a urgencias?
InErUniqueBedCode	Código único de cama en urgencias
InErLocationIdentifier	Descripción corta unidad de enfermería en urgencias
InErNursingUnit	Código de unidad de enfermería en urgencias
InErNursingUnitIsCritics	Es de críticos el código de unidad de enfermería en urgencias
InHospitalDischargeCode	Motivo de alta en hospitalización
InHospitalDischargeDescription	Descripción del motivo de alta en hospitalización
InHospitalHasSurgeryInterventionRequest	¿Tiene una solicitud de intervención en hospitalización?
InHospitalHasSurgeryInterventionRegister	¿Tiene un registro de intervención en hospitalización?
InHospitalHasSurgeryScheduling	Tiene programado/desprogramado una intervención en hospitalización?
InHospitalHasSurgeryBypass	¿Tiene derivada una intervención en hospitalización?
InHospitalHasSurgeryChangeOfOperatingRoom	¿Tiene un cambio de quirófano en hospitalización?
InHospitalSurgeryInterventionRequestComments	Observaciones en solicitud de intervención en hospitalización
InHospitalSurgeryInterventionRequestOperationType	Tipo de la operación a procesar en solicitud de intervención en hospitalización
InHospitalSurgeryInterventionRequestPriority	Prioridad en solicitud de intervención en hospitalización
InHospitalSurgeryInterventionRequestProcedureCode	Código de procedimiento en solicitud de intervención en hospitalización
InHospitalSurgeryInterventionRequestProcedureDescription	Descripción del código de procedimiento en solicitud de intervención en hospitalización
InHospitalSurgeryInterventionRequestDescription	Descripción procedimiento en solicitud de intervención en hospitalización
InHospitalSurgerySchedulingSection	Descripción corta sección programación de intervención en hospitalización

InHospitalSurgerySchedulingSectionCode	Código de sección programación de intervención en hospitalización
InHospitalSurgerySchedulingService	Descripción corta servicio programación de intervención en hospitalización
InHospitalSurgerySchedulingServiceCode	Código de servicio programación de intervención en hospitalización
InHospitalSurgerySchedulingSurgeonCode	Código de cirujano programación de intervención en hospitalización
InHospitalSurgerySchedulingSurgeryRoom	Quirófano programación de intervención en hospitalización
InHospitalSurgeryInterventionRegisterElapsedMinutes	Tiempo transcurrido en minutos desde la entrada hasta la salida en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterOperationType	Tipo de la operación a procesar en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterOrganizationCode	Código de la organización en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterPriority	Prioridad en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterProcedureCode	Código procedimiento en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterProcedureDescription	Descripción de procedimiento en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterProcedureSectionCode	Código de sección en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterProcedureServiceCode	Código de servicio en la intervención en hospitalización
InHospitalSurgeryInterventionRegisterProcedureSurgeryRoom	Código de cirujano en la intervención en hospitalización
InErDischargeCode	Motivo de alta en urgencias
InErDischargeDescription	Descripción del motivo de alta en urgencias
InErHasSurgeryInterventionRequest	¿Tiene una solicitud de intervención en urgencias?
InErHasSurgeryInterventionRegister	¿Tiene un registro de intervención en urgencias?
InErHasSurgeryScheduling	Tiene programado/desprogramado una intervención en urgencias?
InErHasSurgeryBypass	¿Tiene derivada una intervención en urgencias?
InErHasSurgeryChangeOfOperatingRoom	¿Tiene un cambio de quirófano en urgencias?
InErSurgeryInterventionRequestComments	Observaciones en solicitud de intervención en urgencias
InErSurgeryInterventionRequestOperationType	Tipo de la operación a procesar en solicitud de intervención en urgencias
InErSurgeryInterventionRequestPriority	Prioridad en solicitud de intervención en urgencias
InErSurgeryInterventionRequestProcedureCode	Código de procedimiento en solicitud de intervención en urgencias
InErSurgeryInterventionRequestProcedureDescription	Descripción del código de procedimiento en solicitud de intervención en urgencias
InErSurgeryInterventionRequestDescription	Descripción procedimiento en solicitud de intervención en urgencias
InErSurgerySchedulingSection	Descripción corta sección programación

	de intervención en urgencias
InErSurgerySchedulingSectionCode	Código de sección programación de intervención en urgencias
InErSurgerySchedulingService	Descripción corta servicio programación de intervención en urgencias
InErSurgerySchedulingServiceCode	Código de servicio programación de intervención en urgencias
InErSurgerySchedulingSurgeonCode	Código de cirujano programación de intervención en urgencias
InErSurgerySchedulingSurgeryRoom	Quirófano programación de intervención en urgencias
InErSurgeryInterventionRegisterElapsedMinutes	Tiempo transcurrido en minutos desde la entrada hasta la salida en la intervención en urgencias
InErSurgeryInterventionRegisterOperationType	Tipo de la operación a procesar en la intervención en urgencias
InErSurgeryInterventionRegisterOrganizationCode	Código de la organización en la intervención en urgencias
InErSurgeryInterventionRegisterPriority	Prioridad en la intervención en urgencias
InErSurgeryInterventionRegisterProcedureCode	Código procedimiento en la intervención en urgencias
InErSurgeryInterventionRegisterProcedureDescription	Descripción de procedimiento en la intervención en urgencias
InErSurgeryInterventionRegisterProcedureSectionCode	Código de sección en la intervención en urgencias
InErSurgeryInterventionRegisterProcedureServiceCode	Código de servicio en la intervención en urgencias
InErSurgeryInterventionRegisterProcedureSurgeryRoom	Código de cirujano en la intervención en urgencias
InHospitalTransfers	Histórico de traslados en hospitalización
InErTransfers	Histórico de traslados en urgencias
InHospitalTotalElapsedHoursInCritics	Tiempo total en horas en unidades críticas en hospitalización
InHospitalTotalElapsedHoursNonCritics	Tiempo total en horas en unidades no críticas en hospitalización
InHospitalNursingUnit9001ElapsedHours	Tiempo total en horas en unidad 9001 en hospitalización
InHospitalNursingUnit9100ElapsedHours	Tiempo total en horas en unidad 9100 en hospitalización
InHospitalNursingUnit9101ElapsedHours	Tiempo total en horas en unidad 9101 en hospitalización
InHospitalNursingUnit9103ElapsedHours	Tiempo total en horas en unidad 9103 en hospitalización
InHospitalNursingUnit9104ElapsedHours	Tiempo total en horas en unidad 9104 en hospitalización
InHospitalNursingUnit9120ElapsedHours	Tiempo total en horas en unidad 9120 en hospitalización
InHospitalNursingUnit9121ElapsedHours	Tiempo total en horas en unidad 9121 en hospitalización
InHospitalNursingUnit9122ElapsedHours	Tiempo total en horas en unidad 9122 en hospitalización
InHospitalNursingUnit9123ElapsedHours	Tiempo total en horas en unidad 9123 en hospitalización
InHospitalNursingUnit9160ElapsedHours	Tiempo total en horas en unidad 9160 en hospitalización

InHospitalNursingUnit9161ElapsedHours	Tiempo total en horas en unidad 9161 en hospitalización
InHospitalNursingUnit9196ElapsedHours	Tiempo total en horas en unidad 9196 en hospitalización
InHospitalNursingUnit9201ElapsedHours	Tiempo total en horas en unidad 9201 en hospitalización
InHospitalNursingUnit9202ElapsedHours	Tiempo total en horas en unidad 9202 en hospitalización
InHospitalNursingUnit9203ElapsedHours	Tiempo total en horas en unidad 9203 en hospitalización
InHospitalNursingUnit9204ElapsedHours	Tiempo total en horas en unidad 9204 en hospitalización
InHospitalNursingUnit9205ElapsedHours	Tiempo total en horas en unidad 9205 en hospitalización
InHospitalNursingUnit9206ElapsedHours	Tiempo total en horas en unidad 9206 en hospitalización
InHospitalNursingUnit9250ElapsedHours	Tiempo total en horas en unidad 9250 en hospitalización
InHospitalNursingUnit9251ElapsedHours	Tiempo total en horas en unidad 9251 en hospitalización
InHospitalNursingUnit9284ElapsedHours	Tiempo total en horas en unidad 9284 en hospitalización
InHospitalNursingUnit9292ElapsedHours	Tiempo total en horas en unidad 9292 en hospitalización
InHospitalNursingUnit9295ElapsedHours	Tiempo total en horas en unidad 9295 en hospitalización
InHospitalNursingUnit9296ElapsedHours	Tiempo total en horas en unidad 9296 en hospitalización
InHospitalNursingUnit9306ElapsedHours	Tiempo total en horas en unidad 9306 en hospitalización
InHospitalNursingUnit9307ElapsedHours	Tiempo total en horas en unidad 9307 en hospitalización
InHospitalNursingUnit9312ElapsedHours	Tiempo total en horas en unidad 9312 en hospitalización
InHospitalNursingUnit9313ElapsedHours	Tiempo total en horas en unidad 9313 en hospitalización
InHospitalNursingUnit9319ElapsedHours	Tiempo total en horas en unidad 9319 en hospitalización
InHospitalNursingUnit9400ElapsedHours	Tiempo total en horas en unidad 9400 en hospitalización
InHospitalNursingUnit9401ElapsedHours	Tiempo total en horas en unidad 9401 en hospitalización
InHospitalNursingUnit9402ElapsedHours	Tiempo total en horas en unidad 9402 en hospitalización
InHospitalNursingUnit9403ElapsedHours	Tiempo total en horas en unidad 9403 en hospitalización
InHospitalNursingUnit9404ElapsedHours	Tiempo total en horas en unidad 9404 en hospitalización
InHospitalNursingUnit9405ElapsedHours	Tiempo total en horas en unidad 9405 en hospitalización
InHospitalNursingUnit9406ElapsedHours	Tiempo total en horas en unidad 9406 en hospitalización
InHospitalNursingUnit9413ElapsedHours	Tiempo total en horas en unidad 9413 en hospitalización
InHospitalNursingUnit9423ElapsedHours	Tiempo total en horas en unidad 9423 en hospitalización

InHospitalNursingUnit9433ElapsedHours	Tiempo total en horas en unidad 9433 en hospitalización
InHospitalNursingUnit9467ElapsedHours	Tiempo total en horas en unidad 9467 en hospitalización
InHospitalNursingUnit9483ElapsedHours	Tiempo total en horas en unidad 9483 en hospitalización
InHospitalNursingUnit9493ElapsedHours	Tiempo total en horas en unidad 9493 en hospitalización
InHospitalNursingUnit9496ElapsedHours	Tiempo total en horas en unidad 9496 en hospitalización
InHospitalNursingUnit9501ElapsedHours	Tiempo total en horas en unidad 9501 en hospitalización
InHospitalNursingUnit9502ElapsedHours	Tiempo total en horas en unidad 9502 en hospitalización
InHospitalNursingUnit9505ElapsedHours	Tiempo total en horas en unidad 9505 en hospitalización
InHospitalNursingUnit9506ElapsedHours	Tiempo total en horas en unidad 9506 en hospitalización
InHospitalNursingUnit9511ElapsedHours	Tiempo total en horas en unidad 9511 en hospitalización
InHospitalNursingUnit9521ElapsedHours	Tiempo total en horas en unidad 9521 en hospitalización
InHospitalNursingUnit9546ElapsedHours	Tiempo total en horas en unidad 9546 en hospitalización
InHospitalNursingUnit9576ElapsedHours	Tiempo total en horas en unidad 9576 en hospitalización
InHospitalNursingUnit9580ElapsedHours	Tiempo total en horas en unidad 9580 en hospitalización
InHospitalNursingUnit9586ElapsedHours	Tiempo total en horas en unidad 9586 en hospitalización
InHospitalNursingUnit9591ElapsedHours	Tiempo total en horas en unidad 9591 en hospitalización
InHospitalNursingUnit9593ElapsedHours	Tiempo total en horas en unidad 9593 en hospitalización
InHospitalNursingUnit9606ElapsedHours	Tiempo total en horas en unidad 9606 en hospitalización
InHospitalNursingUnit9620ElapsedHours	Tiempo total en horas en unidad 9620 en hospitalización
InHospitalNursingUnit9701ElapsedHours	Tiempo total en horas en unidad 9701 en hospitalización
InHospitalNursingUnit9702ElapsedHours	Tiempo total en horas en unidad 9702 en hospitalización
InHospitalNursingUnit9703ElapsedHours	Tiempo total en horas en unidad 9703 en hospitalización
InHospitalNursingUnit9704ElapsedHours	Tiempo total en horas en unidad 9704 en hospitalización
InHospitalNursingUnit9705ElapsedHours	Tiempo total en horas en unidad 9705 en hospitalización
InHospitalNursingUnit9706ElapsedHours	Tiempo total en horas en unidad 9706 en hospitalización
InHospitalNursingUnit9707ElapsedHours	Tiempo total en horas en unidad 9707 en hospitalización
InHospitalNursingUnit9708ElapsedHours	Tiempo total en horas en unidad 9708 en hospitalización
InHospitalNursingUnit9709ElapsedHours	Tiempo total en horas en unidad 9709 en hospitalización

InHospitalNursingUnit9710ElapsedHours	Tiempo total en horas en unidad 9710 en hospitalización
InHospitalNursingUnit9711ElapsedHours	Tiempo total en horas en unidad 9711 en hospitalización
InHospitalNursingUnit9900ElapsedHours	Tiempo total en horas en unidad 9900 en hospitalización
InHospitalNursingUnit9902ElapsedHours	Tiempo total en horas en unidad 9902 en hospitalización
InHospitalNursingUnit9903ElapsedHours	Tiempo total en horas en unidad 9903 en hospitalización
InHospitalNursingUnit9904ElapsedHours	Tiempo total en horas en unidad 9904 en hospitalización
InHospitalNursingUnit9905ElapsedHours	Tiempo total en horas en unidad 9905 en hospitalización
InHospitalNursingUnit9906ElapsedHours	Tiempo total en horas en unidad 9906 en hospitalización
InHospitalNursingUnit9908ElapsedHours	Tiempo total en horas en unidad 9908 en hospitalización
InHospitalNursingUnit9909ElapsedHours	Tiempo total en horas en unidad 9909 en hospitalización
InHospitalNursingUnit9910ElapsedHours	Tiempo total en horas en unidad 9910 en hospitalización
InHospitalNursingUnit9911ElapsedHours	Tiempo total en horas en unidad 9911 en hospitalización
InHospitalNursingUnit9915ElapsedHours	Tiempo total en horas en unidad 9915 en hospitalización
InHospitalNursingUnit9916ElapsedHours	Tiempo total en horas en unidad 9916 en hospitalización
InHospitalNursingUnit9919ElapsedHours	Tiempo total en horas en unidad 9919 en hospitalización

Para almacenar los datos se ha empleado la tecnología de Microsoft Entity Framework[12], ya que como cualquier ORM[13], permite acceder a una base de datos utilizando clases que representan cada una de las entidades de ésta, pudiendo realizar cualquier operación sobre los datos simplemente llamando a métodos de estas clases.

En la Figura 27 se puede observar que se ha añadido una nueva capa a la solución para trabajar con los datos en una base de datos, en este caso en MySQL[14].

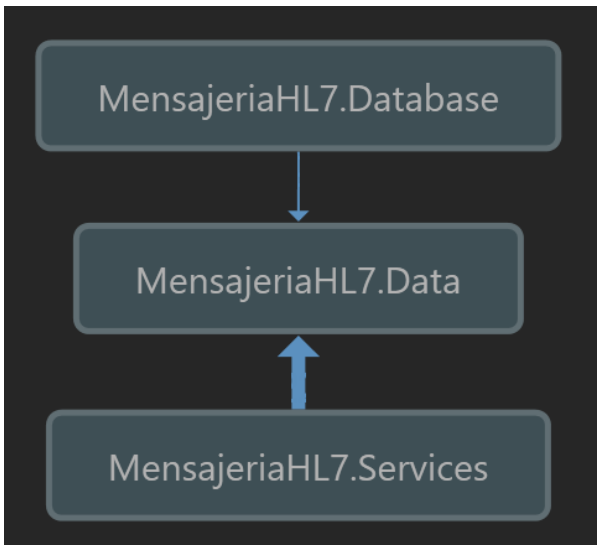
[12] El Entity Framework es un conjunto de tecnologías de ADO.NET que admiten el desarrollo de aplicaciones de software orientadas a datos. Con Entity Framework, los desarrolladores pueden trabajar en un nivel más alto de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales.

[13] Un ORM (Object Relational Mapping) es un tipo de biblioteca de acceso a datos que intenta hacer que esta tarea sea más natural para los desarrolladores. Así, a la hora de acceder a datos, en lugar de utilizar otro lenguaje (generalmente SQL), un ORM permite que puedas utilizar los paradigmas habituales de la programación orientada a objetos: clases y objetos. En lugar de pensar en tablas y relaciones, piensas en objetos y propiedades.

[14] MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto

Figura 27

Definición de la clase circuito



En esta nueva capa se trabaja con la clase MensajeriaHL7Context, que es la encargada de consultar, insertar, actualizar y eliminar datos mediante objetos .NET.

Esta clase hereda de System.Data.Entity.DbContext (denominada clase de contexto). Puede usar un DbContext asociado a un modelo para:

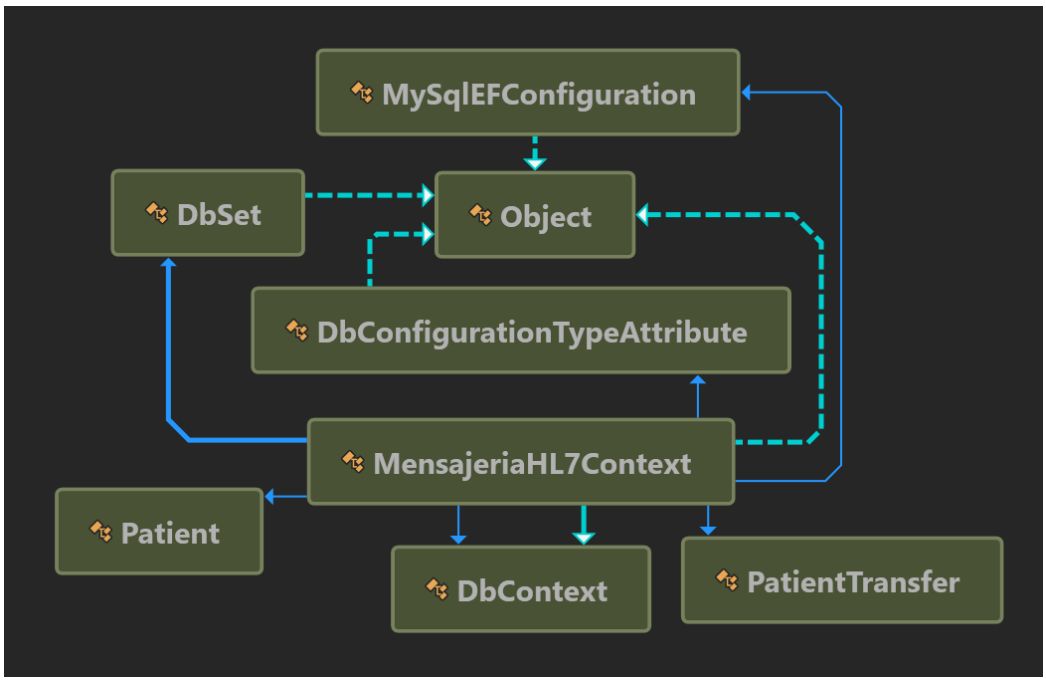
- Escritura y ejecución de consultas
- Materialización de los resultados de la consulta como objetos de entidad
- Seguimiento de los cambios realizados en esos objetos
- Conservar los cambios de objeto en la base de datos
- Enlazar objetos en memoria a controles de interfaz de usuario

En la Figura 28 se muestra el diagrama de dependencias y se pueden ver las dos clases (Patient, dataset A y PatientTransfer, dataset B) que se emplean para rellenar los dos datasets que se emplean en este trabajo.

más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, todo para entornos de desarrollo web.

Figura 28

Diagrama de dependencias de la clase MensajeriaHL7Context

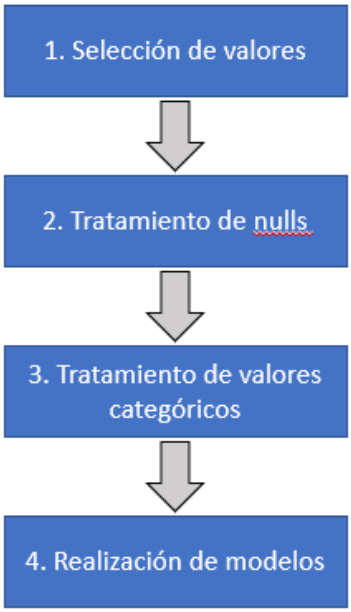


4.3.2 Creación de modelos.

La creación de modelos predictivos no consiste únicamente en implementar los algoritmos matemáticos sobre los datos; sino que para llegar a ese punto hay que realizar una serie de fases previas, tal y como se muestran en la Figura 29:

Figura 29

Fases para la generación de los modelos



1. Selección de valores

En este primer paso, se escogen las variables que después van a ser utilizadas por los modelos. Además, se filtran algunos registros que incumplen algunas reglas básicas. Dentro de todas las columnas disponibles (ver Tabla 5); hay algunas de ellas de las que no es posible extraer información de valor, tanto porque son descripciones de códigos ya presentes en el dataset o porque son campos introducidos por transcripciones manuales de los médicos y para extraer valor de estas últimas sería necesario hacer un análisis del texto escrito mediante técnicas de procesamiento del lenguaje natural (NLP).

Por lo tanto, lo primero que se hace una vez leído el dataset previamente creado es borrar esas columnas que no van a aportar una información válida para los modelos (ver Figura 30).

Figura 30

Borrado de columnas (ejemplo predicción próximo paso alta)

```
##df_base: tabla de partida

##se borran las columnas que no valen para realizar los modelos
df_base = df_base.drop(columns=[
    'Id',
    'CIC',
    'InHospitalDiagnosisDescription',
    'InHospitalTriageRelevantClinicalInformation',
    'InHospitalTriageObservationIdentifierDescription',
    'InHospitalLocationIdentifier',
    'InHospitalDischargeDescription',
    'InHospitalSurgeryInterventionRequestComments',
    'InHospitalSurgeryInterventionRequestProcedureDescription',
    'InHospitalSurgeryInterventionRequestDescription',
    'InHospitalSurgeryInterventionRequestDate', ##no viene bien informado se quita
    'InHospitalSurgerySchedulingScheduledDate', ##no viene bien informado se quita
    'InHospitalSurgerySchedulingSection',
    'InHospitalSurgerySchedulingService',
    'InHospitalSurgeryInterventionRegisterProcedureDescription',
    'InHospitalDischargeCode', ##al predecir el alta no tiene sentido tener esta variable
    'InHospitalTransfers' ## ya hay una variable(InHospitalNursingUnitSteps)que tiene
                          ## los datos de este campo (calculada en el proceso)
])
```

Entre las columnas que han sido seleccionadas pueden darse valores que no deberían de ocurrir en ningún caso; pero, debido a errores en la información interna de los mensajes o en el tratamiento de los mismos, se podrían llegar a observar. Para que no estropeen el resto de la información y hagan que los modelos funcionen de manera anómala esos registros se eliminan. Para ello, se filtran las filas que contienen valores temporales inferiores a cero y se quitan de las tablas de trabajo (ver Figura 31).

Figura 31

Filtrado de valores

```
##obtengo el nombre de todas las columnas cuyo nombre contenga el string 'ElapsedHours'
columns_hour =  df_base.columns[df_base.columns.str.endswith('ElapsedHours')]
```

```
##elimino los registro temporales cuyo valor sea inferior a cero
for c in columns_hour:
    df_base = df_base[df_base[c] >= 0]

df_base = df_base[df_base.InHospitalSurgeryInterventionRegisterElapsedMinutes >= 0]
```

Con estos cambios realizados el número de columnas y de registros del dataset original se ve afectado, pasando de **172** a **175 columnas** en el caso de la estimación de alta por muerte en base al circuito completo hospitalario de los pacientes, y de **13** (únicamente campos de hospitalización) a **117 columnas** para la predicción de alta en base a los traslados que sufre una persona hospitalizada.

2. Tratamiento de nulls

Los valores missing o nulos suponen una perdida significativa de la representatividad de los datos y la mayoría de los modelos predictivos no son capaces de manejarlos. En este trabajo se establecen una serie de estrategias dependiendo de la columna a tratar para solventar este problema.

2.1. Eliminación de registros nulos: Aunque no se dan casos dentro de los datasets que se han estudiado, cuando se encuentra un valor null dentro de los datos personales del paciente (edad y género), se procede a la eliminación de la fila que contiene el mismo (ver Figura 32), ya que se trata de columnas sobre las que no hay un valor para intercambiar y toman una gran importancia en la parte descriptiva del paciente.

Figura 32

Eliminación de registros cuando aparece valor null

```
##casos en los que eliminar el registro cuando aparece un valor null
df_null = df_null.dropna(subset=["Age"])
df_null = df_null.dropna(subset=["Gender"])
```

2.2. Sustitución por un 'No corresponde': Como se puede ver en la tabla que describe los datasets con los que se trabaja en este estudio (ver Tabla 5), hay algunos campos que definen códigos de secciones, operaciones o enfermerías por lo que el paciente ha estado presente. Estas columnas vienen sin informar cuando el paciente no pasa por ninguna de las categorías que ofrece el campo, por ejemplo, si el paciente no ha pasado por urgencias no va a tener información en el campo de la unidad de enfermería de urgencias. En estos casos, se decide crear una nueva categoría la cual se llama 'No corresponde' que sustituye a los valores sin información (ver Figura 33).

Figura 33

Ejemplo sustitución de nulls por 'No corresponde'

```
##selecciono el nombre de las columnas que terminen por los siguientes strings:
columns_code = df_base.columns[df_base.columns.str.endswith('Code')]
columns_room =  df_base.columns[df_base.columns.str.endswith('Room')]
columns_prior =  df_base.columns[df_base.columns.str.endswith('Priority')]
columns_type =  df_base.columns[df_base.columns.str.endswith('Type')]
columns_hour =  df_base.columns[df_base.columns.str.endswith('ElapsedHours')]

##selecciono el nombre de las columnas que contengan los siguientes strings:
columns_timeslot = df_base.columns[df_base.columns.str.contains(pat = 'TimeSlot')]
```

```
##sustituir valores null de ---Code
for c in columns_code:
    df_null[c] = df_null[c].fillna("NoCorresponde").astype(str)
```

```
##sustituir valores null de ---Room
for c in columns_room:
    df_null[c] = df_null[c].fillna("NoCorresponde").astype(str)
```

```
##sustituir valores de ---Type
for c in columns_type:
    df_null[c] = df_null[c].fillna("NoCorresponde").astype(str)
```

2.3. Sustitución por 0: Dentro de la información disponible para realizar las predicciones, hay muchas columnas que informan del tiempo en horas o minutos que pasa un paciente en una ubicación determinada. Estos campos se encuentran siempre informados, pero en el caso de que no sea así y viniese un 'null' o valor nulo se tomaría la consideración de que esa persona no ha pasado por la unidad o sección, por lo que ese registro se informaría a 0. Del mismo modo, entre los datos con los que se ha trabajado también hay campos binarios que marcan, por ejemplo, si un paciente ha sido hospitalizado o si tiene alergias. Si esos datos se encuentran sin informar se considerará que no cumple la condición marcada por la columna por lo que se sustituirá el valor 'null' por 0 (ver Figura 34).

Figura 34

Ejemplo sustitución de nulls por 0

```
##sustituir valores de null ---ElapsedHours
for c in columns_hour:
    df_null[c] = df_null[c].fillna(0)
```

```
##tratamiento de nulls sobre columnas individualizadas:
df_null['InHospitalNursingUnitIsCritics'] = df_null['InHospitalNursingUnitIsCritics'].fillna(0)

df_null['InHospitalHasSurgeryBypass'] = df_null['InHospitalHasSurgeryBypass'].fillna(0)

df_null['InHospitalHasSurgeryScheduling'] = df_null['InHospitalHasSurgeryScheduling'].fillna(0)

df_null['InHospital'] = df_null['InHospital'].fillna(0)

df_null['WithAllergies'] = df_null['WithAllergies'].fillna(0)
```

3. Tratamiento de valores categóricos

Cuando se realiza un modelo predictivo hay que tener en cuenta que se pueden tener variables numéricas o categóricas. En el caso de estudio de este trabajo se encuentran valores de ambos tipos. Las variables numéricas son aquellas que pueden tomar cualquier valor numérico real, por ejemplo, el tiempo total en horas en la unidad 9903 de hospitalización (*InHospitalNursingUnit9903ElapsedHours*); mientras que las variables categorías son aquellas que utilizan un número limitado de valores distintos, por ejemplo, los motivos de alta en emergencias (*InErDischargeCode,* ver Figura 35).

Figura 35

Ejemplo variable categórica [InErDischargeCode]

InErDischargeCode	InErDischargeDescription
6	Fallecido
10	Alta por ingreso
1	Alta médica domicilio habitual
7	Alta voluntaria
9	Otras causas
4	Traslado a media y larga estancia
2	Alta médica hospit. domicilio
3	Traslado a hospital de agudos
5	Alta médica residencia social

Muchos de los algoritmos de predicción no son capaces de trabajar con variables categóricas, por lo que hay que cambiar esos datos a numéricos, para ello se siguen las siguientes estrategias; ambas han sido empleadas en el trabajo y comparadas, también, en los resultados de los modelos.

3.1. **Label encoding:** En esta técnica a cada etiqueta o clase de la columna a codificar se le asigna un número único entero ordenado de manera alfabética. Esta codificación es muy sencilla, pero tiene el problema de que algunos algoritmos pueden interpretarlo de manera errónea considerando relaciones entre las diferentes clases; por ejemplo, si en la columna InHospitalNursingUnit que define el código de la unidad de enfermería en hospitalización se codifica la unidad ‘9403’ como 26 y la unidad ‘9910’ como 52, el modelo puede considerar que la unidad 9910 es el doble que la 9403, cuando tendría que ser independientes entre ellas (ver Figura 36).

Figura 36

Ejemplo label encoding [InHospitalNursingUnit]

InHospitalNursingUnit	InHospitalNursingUnit_label
9101	0
9103	1
9104	2
9120	3
9121	4
9122	5
9123	6
9160	7
9161	8
9196	9
9201	10
9202	11
9203	12

3.2. **One-hot encoding:** La estrategia que lleva a cabo esta técnica de codificación consiste en la creación de una columna por cada una de las clases de la variable categórica rellenándose todas a cero excepto cuando el valor de la variable categórica coincide con la clase de la columna, en cuyo caso se marca con un uno, creando una variable booleana (ver Figura 37). Este método no conlleva relaciones entre los valores de la variable categórica, pero es menos eficiente ya que el número de columnas de la tabla de trabajo se ve incrementado de manera considerable.

Figura 37

Ejemplo one-hot encoding [InHospitalNursingUnit])

InHospitalNursingUnit	InHospitalNursingUnit_9205	InHospitalNursingUnit_9403	InHospitalNursingUnit_9493	InHospitalNursingUnit_9910
9403	0	1	0	0
9205	1	0	0	0
9402	0	0	0	0
9101	0	0	0	0
9250	0	0	0	0
9910	0	0	0	1
9493	0	0	1	0
9204	0	0	0	0
9916	0	0	0	0
9493	0	0	1	0
9404	0	0	0	0
9403	0	1	0	0
9403	0	1	0	0

4. Realización de modelos:

Una vez los datos están listos, tras aplicar las anteriores transformaciones sobre ellos, se procede a la realización de algoritmos matemáticos con el fin de predecir los valores que se han planteado:

- Alta por muerte de un paciente a partir del circuito completo de los episodios hospitalarios o de urgencias.
- Alta de un paciente (alta hospitalaria, alta voluntaria, muerte, etc.) a partir de los datos del conjunto de traslados hospitalarios de los pacientes.

Ambas predicciones requieren de algoritmos de clasificación con aprendizaje supervisado. La realización de esta tarea no se ha efectuado con un único algoritmo, sino que se han llevado a cabo modelos diferentes y después, en base a las métricas obtenidas, se ha decidido cual es el que mejor encaja en cada problema.

Para implementar los algoritmos se han utilizado las siguientes librerías de Python:

- *Scikit-Learn*: Librería de código abierto utilizada, principalmente, para labores de aprendizaje automático y machine learning; permite la ejecución de una gran variedad de algoritmos, así como la realización de validación cruzada y la extracción de métricas.
- *Keras*: Librería de código abierto, al igual que *Scikit-Learn,* que facilita la creación de redes neuronales, no funciona como un marco de trabajo independiente, sino que puede ejecutarse sobre diferentes plataformas como pueden ser *CNTK*, *Theano* o *Tensorflow*. En este trabajo, se utiliza para construir una red neuronal empleando la sistemática de bloques que permite definir cada capa que compone el modelo.
- *Tensorflow*: Biblioteca de código abierto, empleada por *Keras*, usada también para el aprendizaje automático, que destaca, principalmente, en el desarrollo y entrenamiento de redes neuronales.
- *Imbalanced Learn:* Librería de código abierto, basada en *Scikit-Learn*, que proporciona herramientas para trabajar con datos desbalanceados.

Además de la realización de los modelos, se han desarrollado diferentes técnicas para obtener los parámetros óptimos que mejor resultado dan para cada algoritmo. En el punto 5. Desarrollo de los modelos predictivosde la memoria se detallan los diferentes modelos empleados, la configuración que se ha utilizado y las estrategias llevadas a cabo para obtener dichas configuraciones.

4.3.3 Obtención de resultados.

Una parte clave del trabajo desarrollado es el análisis de los resultados obtenidos en los modelos. Para entender las métricas que se han empleado es importante detallar, previamente, estos conceptos:

- Verdaderos positivos (**TP**): Número de casos que la predicción declara positivo y son verdaderamente positivos.
- Verdaderos negativos (**TN**): Número de casos que la predicción declara negativo y son verdaderamente negativos.
- Falsos positivos (**FP**): Número de casos que la predicción declara positivo y son verdaderamente negativos.
- Falsos negativos (**FN**): Número de casos que la predicción declara negativo y son verdaderamente positivos.

Este estudio realiza una serie de métricas que permiten evaluar el rendimiento de los algoritmos; estas métricas son:

1. **Accuracy:** especifica el porcentaje de predicciones verdaderas, es decir, aquellas que el modelo ha acertado. Representa el número de casos correctos frente al total:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. **Especificidad:** Mide los casos negativos que el algoritmo ha clasificado correctamente.

$$especificidad = \frac{TN}{TN + FP}$$

3. **Recall/sensibilidad:** Mide la proporción de casos positivos que fueron correctamente identificadas por el algoritmo.

$$sensibilidad = \frac{TP}{TP + FN}$$

4. **Matriz de confusión:** Esta matriz se representa siempre en forma de tabla, de manera que en cada columna aparece el número de predicciones de cada clase, mientras que cada fila muestra el número real de instancias de cada clase (ver Figura 38). Se utiliza para ver el desempeño general del modelo.

Figura 38

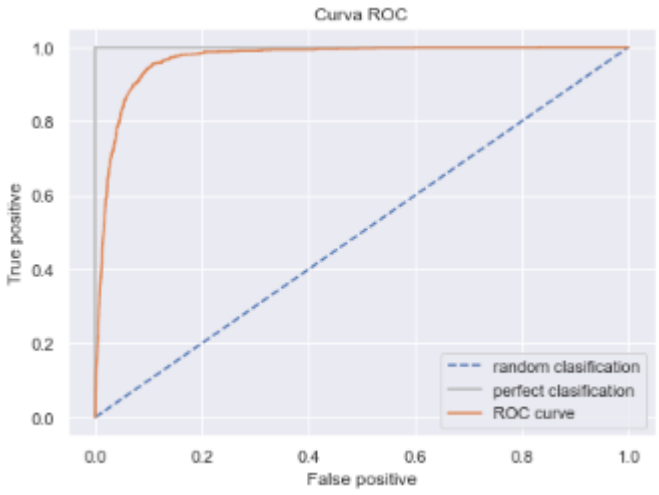
Ejemplo de matriz de confusión

		Predicción	
		Negativos	Positivos
Valor real	Negativos	TN	FP
	Positivos	FN	TP

5. **Curva ROC (Receiver Operating Characteristic):** Indica la ratio de positivos reales versus la ratio de positivos falsos en distintos umbrales de clasificación (ver Figura 39). Sirve para mostrar, de forma explícita, cuándo una clase es confundida con otra.

Figura 39

Ejemplo de curva ROC



6. **AUC (Área bajo la curva ROC):** Se trata de un simple cálculo integral de la superficie que se encuentra debajo de la curva del modelo. Un modelo perfecto tendría un AUC de 1, mientras que un modelo pésimo tendría un valor de 0.

5. Desarrollo de los modelos predictivos

En este apartado se detalla el proceso efectuado para realizar los modelos de aprendizaje automático e inteligencia artificial que han sido aplicados en el trabajo. Del mismo modo, se especifican las estrategias que se han seguido para configurar estos algoritmos de manera apropiada, explicando en qué consisten estas técnicas y qué proporcionan a los algoritmos.

Las clases que se predicen en estos modelos están claramente desbalanceadas; como se puede observar en la Figura 40 y la Figura 41, las altas por muerte son significativamente menores al resto de casos que suponen el fin de un episodio hospitalario. Del mismo modo, el hecho de ser dado de alta dentro de un conjunto de traslados es inferior, ya que normalmente cuando se es hospitalizado se recorren una serie de unidades antes de finalizar la estancia.

Figura 40

Distribuciones de las clases. Predicción de alta

False	15675
True	3977

Figura 41

Distribuciones de las clases. Predicción fallecimiento

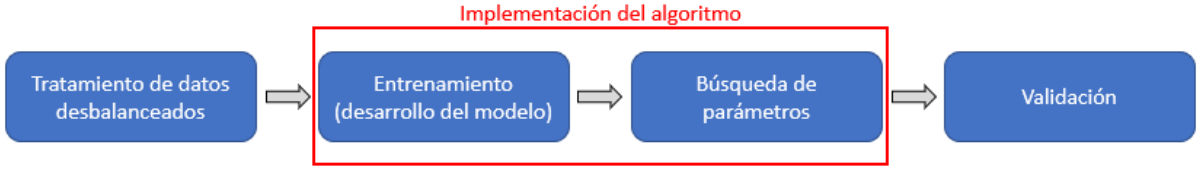
False	55893
True	948

Para tratar estos datos desbalanceados se han utilizado diferentes técnicas con el fin de intentar igualar ambas clases lo máximo posible y mejorar el rendimiento de los modelos realizados.

En la Figura 42 Esquema desarrollo de modelos se muestra la estrategia que se ha seguido para implementar los algoritmos con los que se ha trabajado. Algunas fases de este esquema no son necesarias en todos los modelos, lo cual se detallará cuando se explique cada uno de forma individualizada.

Figura 42

Esquema desarrollo de modelos



Cuando se desarrolla un modelo, lo primero que se realiza es el tratamiento de los datos desbalanceados, tras aplicar las técnicas seleccionas, se procede a la implementación del algoritmo que consiste en la definición del modelo y la búsqueda de los parámetros que permitan un mejor funcionamiento de este; por último, se establecen las validaciones oportunas para comprobar el correcto funcionamiento de todo los desarrollado hasta este punto.

5.1 Tratamiento de datos desbalanceados

Como se ha indicado anteriormente, se han planteado experimentos donde la clase a predecir esta desbalanceada. Esto supone un problema ya que los algoritmos tienden a generalizar y les resulta complicado estimar adecuadamente las clases minoritarias. Para intentar solventar esto, se establece un par de estrategias dependiendo de la predicción a realizar, ya que el nivel de desbalanceo es diferente tal y como se aprecian en la Figura 40 y la Figura 41:

A) En el experimento que estima el alta de un paciente, se realiza un oversampling (sobremuestreo) sobre la clase minoritaria con el objetivo de igualar ambas. Para realizar ese aumento de registros de la clase con menor representación se aplica *SMOTE*, método que se basa en el algoritmo kNN (k-Nearest Neighbour), que permite crear nuevos registros con características similares a los de la clase a incrementar.

   En la Figura 43 se observa un ejemplo de su implementación, viendo cómo se igualan las clases tras aplicarlo.

   Figura 43

   Código sobremuestreo en estimación de alta

```
##miro el conteo de clases de la variable a predecir en el entrenamiento
y_label.iloc[train_index].value_counts()
```

```
False    12540
True      3182
Name: InHospitalNursingUnitNext, dtype: int64
```

```
##aplico oversampling
sm = SMOTENC(random_state=42, categorical_features=cat_indices_label)
X_train_over_label, y_train_over_label = sm.fit_resample(x_label.iloc[train_index], y_label.iloc[train_index])
```

```
##miro los resultados del oversampling
y_train_over_label.value_counts()
```

```
True     12540
False    12540
Name: InHospitalNursingUnitNext, dtype: int64
```

B) En el experimento que estima el fallecimiento de un paciente, la técnica empleada es algo más compleja, ya que el desbalanceo entre clases es mayor. En este caso, no consiste únicamente en hacer un sobremuestreo, sino que hay que realizar

posteriormente un submuestreo sobre la clase mayoritaria. El aumento de elementos en la clase minoritaria se realiza también con la técnica SMOTE, pero en este caso no se igualan las clases, sino que se opta por dejarla en el 50% de la clase mayoritaria.

Posteriormente, tras aplicar el sobremuestreo, se procede a la reducción del número de ocurrencias de la clase más representada, en este caso se emplea la técnica *TomekLink*, que consiste en borrar únicamente aquellos registros de la clase mayoritaria que estén en zonas donde se ubican las variables minoritarias, permitiendo así reducir el ruido.

En la Figura 44 se observa un ejemplo de su implementación, viendo el número de registros de cada categoría en los diferentes pasos realizados.

Figura 44

Técnicas de desbalanceo en estimación de fallecimiento

```
##miro el conteo de clases de la variable a predecir
y_prueba.value_counts()
```

```
False    55893
True       948
dtype: int64
```

```
##aplico oversampling y miro resultados tras aplicarlo
sm = SMOTENC(random_state=1, categorical_features=cat_indices,sampling_strategy=0.5)

X_prueba_oversampled, y_prueba_oversampled = sm.fit_resample(x_prueba, y_prueba)
y_prueba_oversampled.value_counts()
```

```
False    55893
True     27946
dtype: int64
```

```
##aplico undersampling y miro resultados tras aplicarlo
under = TomekLinks(sampling_strategy='majority')

X_prueba_undersampled, y_prueba_undersampled = under.fit_resample(X_prueba_oversampled, y_prueba_oversampled)
y_prueba_undersampled.value_counts()
```

```
False    55587
True     27946
dtype: int64
```

5.2 Implementación del algoritmo

Tras aplicar las técnicas de desbalanceo, se definen los modelos y se buscan los parámetros de estos que mejor hagan funcionar al algoritmo en cada experimento.

5.2.1 Naive Bayes

Algoritmo de clasificación basado en el Teorema de Bayes, se trata de un modelo simple y rápido, que no requiere de ningún tipo de parámetro para ajustar su comportamiento, por lo que no se aplicará búsqueda de estos.

Inicialmente, se define el algoritmo a utilizar, en este caso Naive Bayes gaussiano, el cual supone que los datos de entrada siguen una distribución gaussiana. Con el modelo ya

definido, se entrena con los datos que han sido seleccionados para esa tarea y se realizan las predicciones tanto categóricas como probabilísticas. En la Figura 45 se observa el código desarrollado para implementar el algoritmo.

Figura 45

Implementación de Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

##creo un clasificador gaussiana
nb_gss = GaussianNB()

##creo variable en las que se almacenan los resultados de cada fold
y_pred = []
y_prob = []
y_test = []
acc = []

##puntero
i = 0

##recorro los diferentes folds creados (en este caso 5) y leo los indices seleccionados:
for train_index, test_index in kf.split(x_label,y_label):

    ##tratamiento de clases desbalanceadas
    sm = SMOTENC(random_state=1, categorical_features=cat_indices_label,sampling_strategy=0.5)
    X_train_over_label, y_train_over_label = sm.fit_resample(x_label.iloc[train_index], y_label.iloc[train_index])
    under = TomekLinks(sampling_strategy='majority')
    X_train_under_label, y_train_under_label = under.fit_resample(X_train_over_label, y_train_over_label)

    print(y_train_under_label.value_counts())

    ##entreno el modelo
    nb_gss.fit(X_train_under_label, y_train_under_label)

    ##realizo la predicción
    y_pred.append(nb_gss.predict(x_label.iloc[test_index]))

    ##realizo la prediccion pobabilísticas (utilizada en la curva ROC)
    y_prob.append(nb_gss.predict_proba(x_label.iloc[test_index]))
```

1

2

3

4

Nota:

1) Definición del algoritmo
2) Tratamiento de datos desbalanceados
3) Entrenamiento del modelo
4) Predicciones

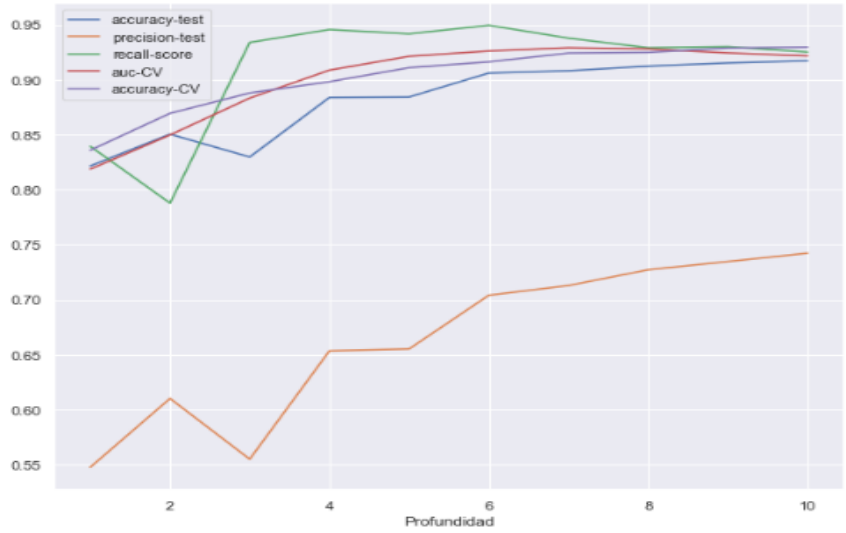
5.2.2 Árboles de decisión

Algoritmo predictivo que, a partir de un conjunto de reglas, permite dividir el espacio de las variables predictoras agrupando observaciones con valores similares para la variable a estimar.

Con el fin de configurar los parámetros del modelo que mejor encajen con los datos de estudio, se evalúa el rendimiento del mismo para diferentes profundidades, midiéndose en cada una las siguientes métricas: accuracy, recall y precisión, y seleccionando aquella que mejor encaja. En la Figura 46 se muestran las métricas recogidas y como se puede observar la profundidad óptima para este árbol es de 6, ya que a partir de ese punto varias métricas empiezan a saturar y a empeorar sus registros.

Figura 46

Métricas obtenidas en los árboles de decisión



Una vez establecidos los parámetros del árbol, se procede a la implementación del mismo. Realizándose primero la definición del modelo, posteriormente el entrenamiento y por último las predicciones sobre el conjunto de test. En la Figura 47 se observa el código desarrollado.

Figura 47

Implementación de árboles de decisión

```
##creo variable en las que se almacenan los resultados de cada fold
y_pred = []
y_prob = []
y_test = []
acc = []

##puntero
i = 0

##creo el arbol de decision con profundidad 6
clf = DecisionTreeClassifier(max_depth=6, min_samples_split=20, criterion='entropy')

##recorro los diferentes folds creados (en este caso 5) y leo los indices seleccionados:
for train_index, test_index in kf.split(x_label,y_label):

    ##oversampling
    sm = SMOTENC(random_state=1, categorical_features=cat_indices_label)
    X_train_oversampled, y_train_oversampled = sm.fit_resample(x_label.iloc[train_index], y_label.iloc[train_index])

    ##entreno el modelo
    clf = clf.fit(X_train_oversampled, y_train_oversampled)

    ##realizo la predicción
    y_pred.append(clf.predict(x_label.iloc[test_index]))

    ##realizo la prediccion pobabilísticas (utilizada en la curva ROC)
    y_prob.append(clf.predict_proba(x_label.iloc[test_index]))
```

1) Definición del algoritmo
2) Oversampling
3) Entrenamiento del modelo
4) Predicciones

1

2

3

4

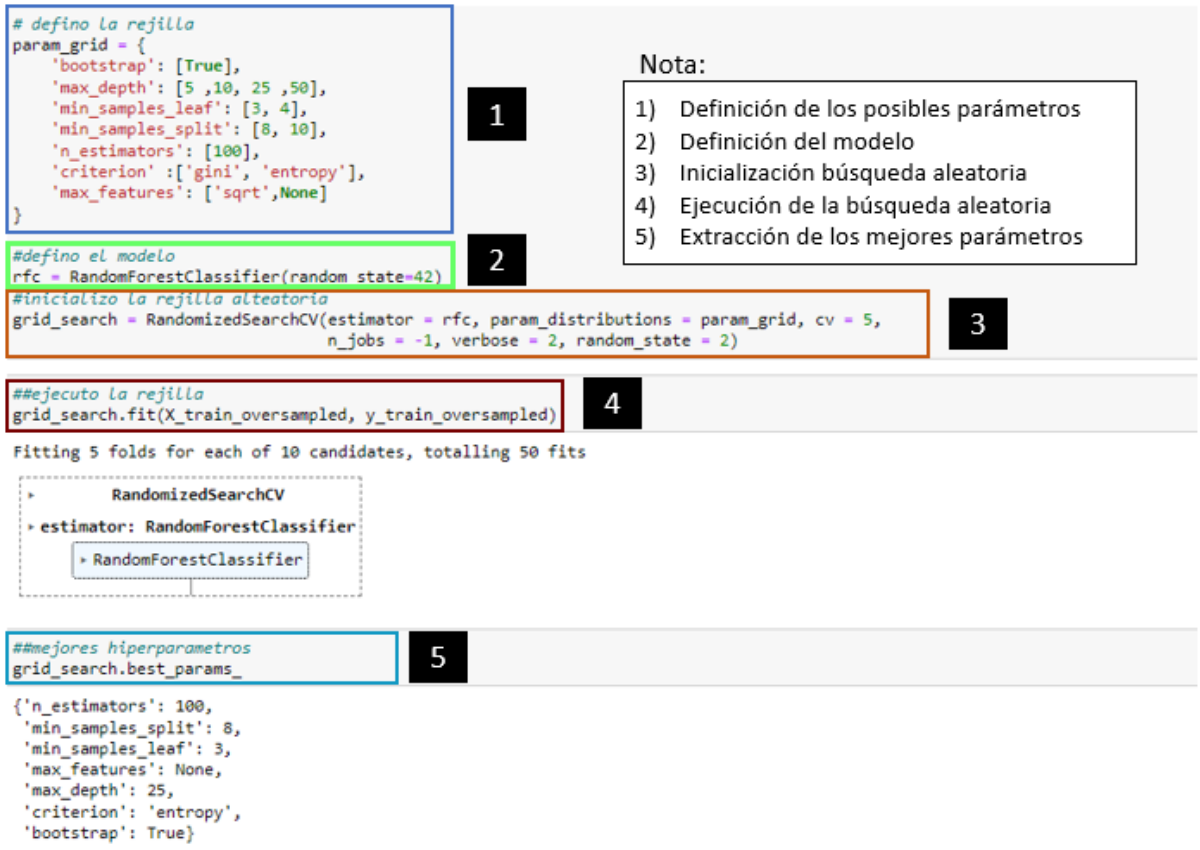
5.2.3. Random Forest

Modelo predictivo, que consiste en la combinación de árboles de decisión, basándose en los principios de bagging y en la selección de variables de forma aleatoria para cada uno de los árboles.

Para ajustar los parámetros del modelo se ha utilizado *RandomizedSearchCV* de *sklearn*, que permite configurar una serie de valores propios del modelo y establecer una serie de opciones para cada uno de ellos. La técnica consiste en realizar una búsqueda aleatoria de hiperparámetros, de manera que muestrea de manera uniforme todas las posibles combinaciones establecidas y selecciona aquella que tiene la mejor métrica, en este caso el *accuracy*. Se ha seleccionado esta funcionalidad frente a la búsqueda cartesiana, porque computacionalmente es bastante más eficiente y el tiempo de ejecución es mucho menor, además que los resultados arrojados al comprobar todas las posibles combinaciones de parámetros no son significativamente mejores. En la Figura 48 se puede observar cómo se ha codificado esta técnica.

Figura 48

Técnica RandomizedSearchCV sobre RandomForest




```
# defino la rejilla
param_grid = {
    'bootstrap': [True],
    'max_depth': [5 ,10, 25 ,50],
    'min_samples_leaf': [3, 4],
    'min_samples_split': [8, 10],
    'n_estimators': [100],
    'criterion' :['gini', 'entropy'],
    'max_features': ['sqrt',None]
}
```

```
#defino el modelo
rfc = RandomForestClassifier(random_state=42)
```

```
#inicializo la rejilla aleatoria
grid_search = RandomizedSearchCV(estimator = rfc, param_distributions = param_grid, cv = 5,
                                 n_jobs = -1, verbose = 2, random_state = 2)
```

```
##ejecuto la rejilla
grid_search.fit(X_train_oversampled, y_train_oversampled)
```

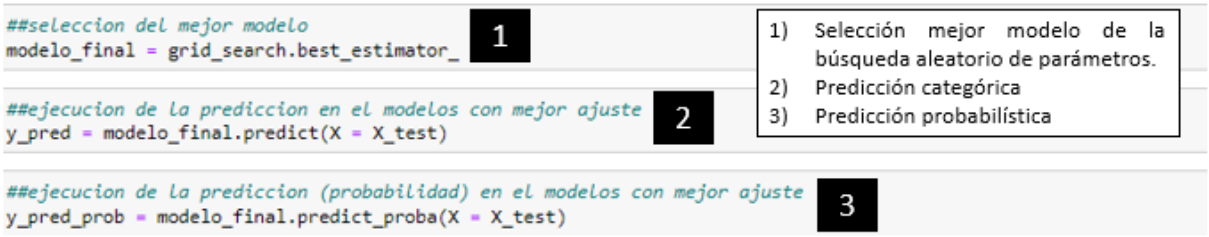
```
##mejores hiperparametros
grid_search.best_params_
```

```
{'n_estimators': 100,
 'min_samples_split': 8,
 'min_samples_leaf': 3,
 'max_features': None,
 'max_depth': 25,
 'criterion': 'entropy',
 'bootstrap': True}
```

Con el mejor modelo seleccionado por la anterior técnica, se realizan las predicciones tanto categóricas como probabilísticas, tal y como se puede ver en la Figura 49, para llevar a cabo esas estimaciones se tiene en cuenta un conjunto de los datos que no se utilizan para entrenar y se reservan para validar.

Figura 49

Realización predicciones RandomForest



5.2.4 Gradient Boosting Decision Tree

Algoritmo que se basa en la combinación de árboles de decisión de forma secuencial, de manera que cada nuevo árbol incorporado utiliza la información del anterior y trata de suplir los errores que ha podido cometer.

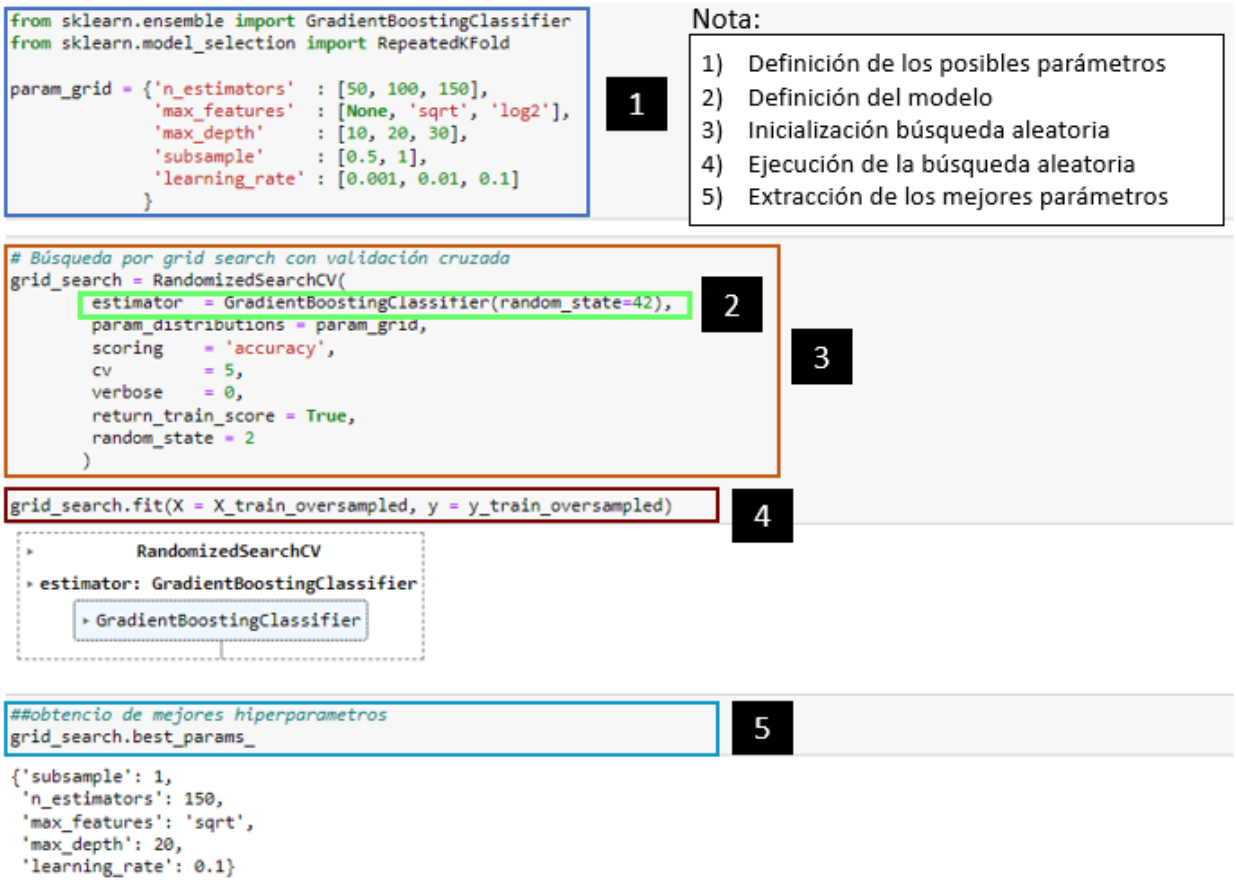
Al igual que para el modelo de *RandomForest*, para ajustar los parámetros que emplea GBDT se utiliza una rejilla aleatoria, más concretamente la funcionalidad *RandomizedSearchCV* de *sklearn*. Aunque en este caso no es totalmente similar al anterior modelo, como se puede ver en la Figura 50, la magnitud de los parámetros seleccionados y la naturaleza de los mismos es diferente, ya que al ser otro modelo los parámetros que actúan sobre él no son similares.

La rejilla de los datos se describe mediante un diccionario, donde las claves definen los nombres de los parámetros dados por la librería para ese modelo, y los valores de los diccionarios son listas con las magnitudes que se quieren probar para cada parámetro definido en las claves (punto 1 de la Figura 50).

Al inicializar *RandomizedSearchCV*, se le da como parámetro la rejilla aleatoria previamente creada. Del mismo modo, también se establece la métrica sobre la que elegir el mejor modelo, en este caso el *accuracy,* y se define la validación y otros valores como el '*verbose'* que dependiendo del valor tenga muestra diferente información por pantalla (punto 2 de la Figura 50).

Figura 50

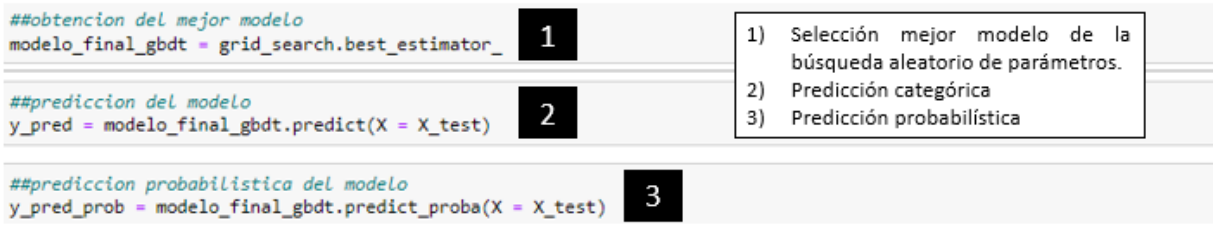
Técnica RandomizedSearchCV sobre Gradient Boosting Decision Tree



Cuando se ha obtenido el mejor modelo con la técnica anterior, se procede a la predicción sobre el conjunto de prueba que se haya establecido. *RandomizedSearchCV* permite seleccionar ese mejor modelo de manera directa, sin tener que hacer la definición del mismo con los mejores hiperparámetros, llamando a la función *'best_estimator_'* tal y como se aprecia en la Figura 51.

Figura 51

Realización predicciones Gradient Boosting Decision Tree



5.2.5 Redes neuronales Perceptrón Multicapa (MLP)

Red neuronal compuesta por una capa de entrada, otra de salida y n capas ocultas entre ambas. Entrenar una red neuronal supone encontrar todos los valores de los pesos y bias de las neuronas que componen las diferentes capas.

La implementación conlleva muchos parámetros a decidir, como son:

1. Número de capas.
2. Cantidad de neuronas en cada capa.
3. Funciones de activación de cada capa.
4. Funciones de perdida y métricas durante la compilación.
5. Numero de épocas y tamaño del bath en el entrenamiento.

El número de neuronas en la capa de entrada se corresponde con la cantidad de variables predictoras; y las neuronas en la capa de salida, con las variables a predecir, en este caso una.

Como función de activación en las capas ocultas se emplea ReLU (Figura 52) ya que, computacionalmente, es la más eficiente y la no existencia de saturación en ésta permite un entrenamiento más ágil. Para la capa de salida, se emplea una función de saturación sigmoid pues mapea los valores finales entre 0 y 1 (resultado binario). Al tratarse de un problema de clasificación binaria durante la compilación se utiliza una función de perdida *binary_crossentropy* y como métrica el accuracy.

Figura 52

Definición arquitectura red neuronal

```
##definicion de la arquitectura de la red neuronal
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(32,activation='relu',input_shape=(167,))) 1
model.add(tf.keras.layers.Dense(16,activation='relu'))                      2
model.add(tf.keras.layers.Dense(8,activation='relu'))
model.add(tf.keras.layers.Dense(1,activation='sigmoid'))                    3
```

1 Tamaño de la capa de entrada (167 variables)

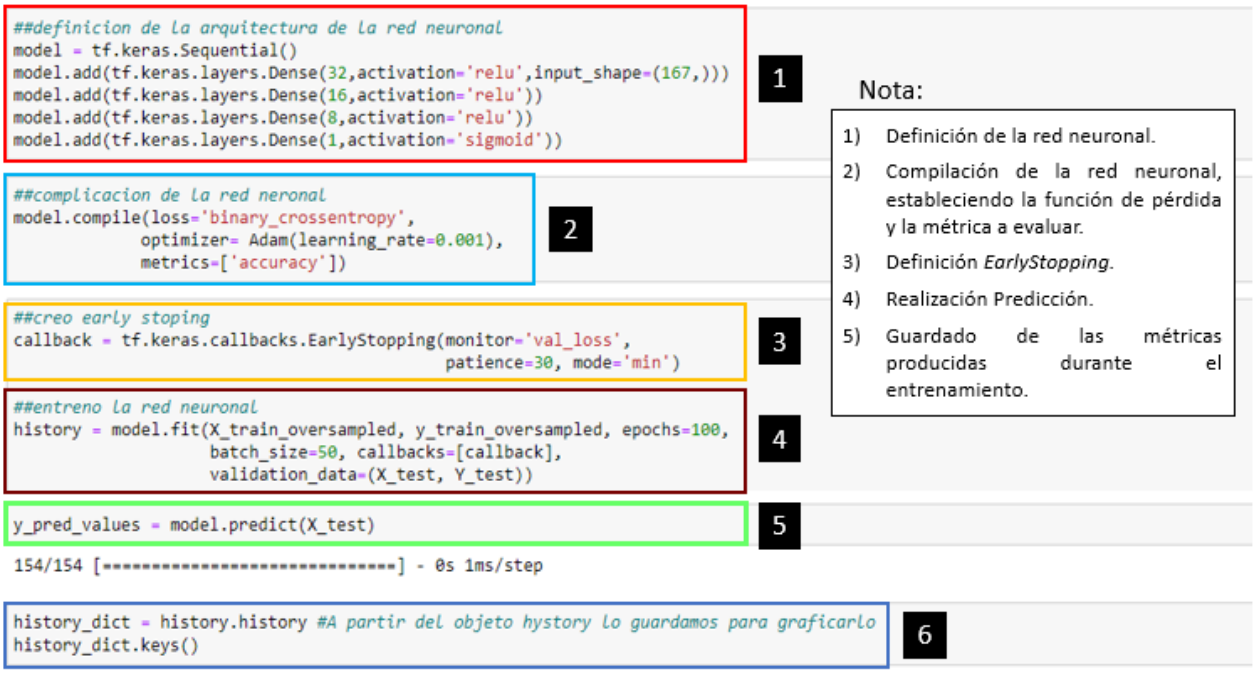
2 Definición de las capas ocultas, donde se establece la función de activación ***ReLU*** y la cantidad de neuronas (32,16 y 8)

3 Definición de la capa de salida, con una única neurona y la función de activación ***sigmoid***

Por otro lado, el resto de los parámetros que se pueden personalizar en una red neuronal, se introducen tras probar diferentes configuraciones y comprobar las que mejor resultado dan al problema. Obteniendo finalmente la siguiente configuración, observar Figura 53, en el caso de la predicción de alta utilizando codificación one-hot.

Figura 53

Implementación red neuronal

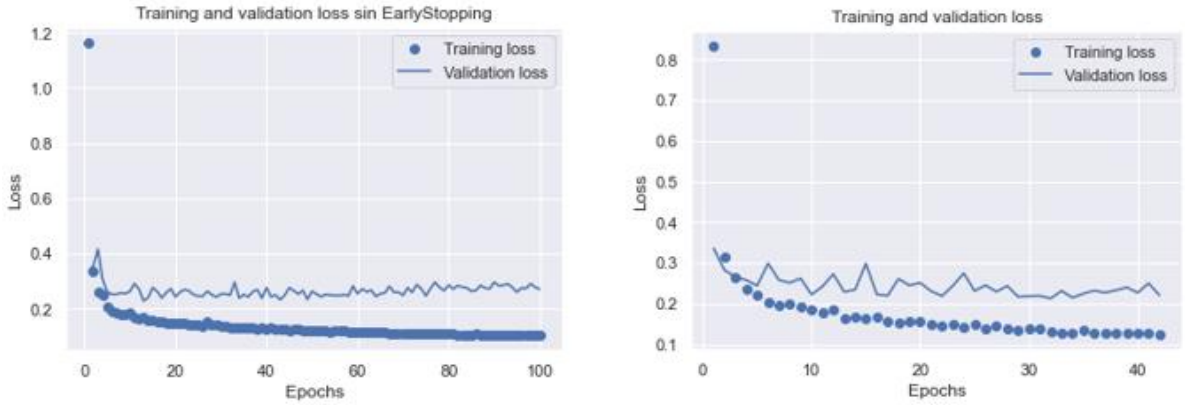


Además, se ha empleado la técnica de *early stopping* que evita el sobre ajuste de la red neuronal a los datos de entrenamiento, parando la fase de aprendizaje de la red neuronal cuando las métricas sobre los datos de validación empiezan a saturar y dejan de arrojar mejores valores que las anteriores fases (epochs).

En la Figura 54, se muestran las métricas obtenidas durante el entramiento para una red neuronal que utiliza *EarlyStopping* (gráfica derecha) y otra que no (gráfica izquierda). Se puede observar como el modelo que no emplea la funcionalidad realiza 100 fases (todas las que tenía definidas) y a partir de la 30 o 40 aumenta levemente su valor de error en los datos de validación y por lo tanto empieza a aparecer el overfitting sobre los datos de entrenamiento. Por otro lado, la red neuronal que emplea *EarlyStopping* tiene también definida 100 fases de entramiento, pero en este caso cuando empieza a aumentar el error de validación se para, habiendo realizado 42 epochs, hay que tener en cuenta que se define una paciencia para parar, en este caso 10, esto quiere decir que se espera 10 epochs a partir del valor mínimo de validación obtenido para ver si sigue decreciendo o aumenta.

Figura 54

Comparativa EarlyStopping



5.3 Validación de los modelos

Para estimar el rendimiento del algoritmo, una vez creado el modelo, se realiza validación cruzada (*cross-validation,* ver Figura 55); esta técnica consiste en dividir los datos en k particiones, en este caso 5, creándose un modelo que utiliza 4 particiones para entrenar y una restante para evaluar; se realiza este proceso 5 veces desarrollándose las diferentes combinaciones posibles, actuando finalmente todas las particiones una vez como validación. Al utilizar unos datos desbalanceados, se requiere el uso de la validación cruzada *Stratified*, que permite mantener la distribución de clases en cada una de las particiones seleccionadas.

Figura 55

Stratified k-folds cross-validation

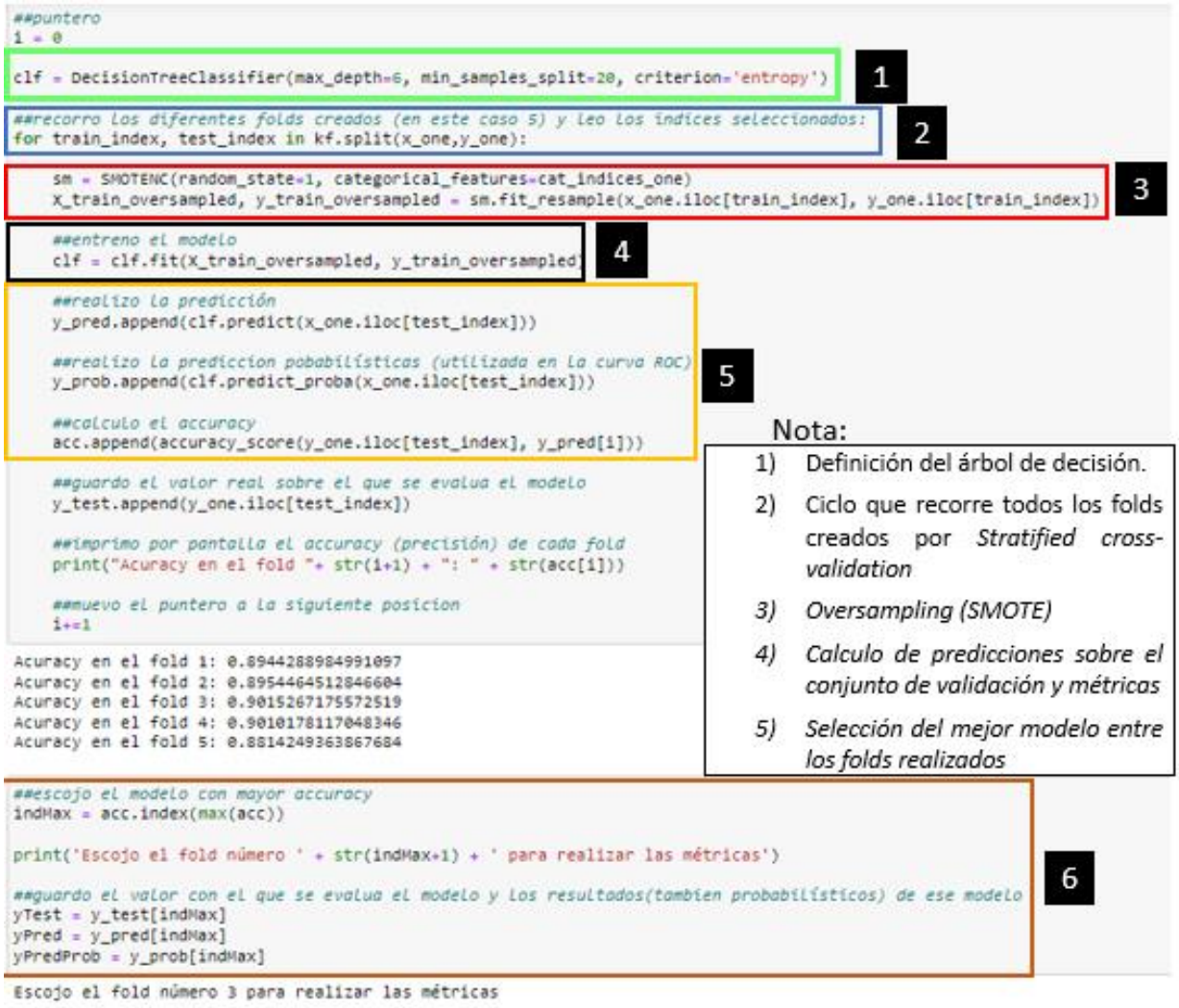
```
from sklearn.model_selection import StratifiedKFold

##Defino la técnica K-fold con K=5
kf = StratifiedKFold(n_splits=5)

##imprimo por pantalla los indices train y text de cada fold
for train_index, test_index in kf.split(x_label,y_label):
    print("Train Index: ", train_index)
    print("Test Index: ", test_index, "\n")
```

```
Train Index:  [ 3857  3863  3886 ... 19649 19650 19651]
Test Index:  [   0    1    2 ... 3941 3942 3945]

Train Index:  [    0     1     2 ... 19649 19650 19651]
Test Index:  [3857 3863 3886 ... 7987 7994 8002]

Train Index:  [    0     1     2 ... 19649 19650 19651]
Test Index:  [ 7828  7829  7830 ... 11830 11831 11832]

Train Index:  [    0     1     2 ... 19649 19650 19651]
Test Index:  [11773 11775 11776 ... 15956 15957 15959]

Train Index:  [    0     1     2 ... 15956 15957 15959]
Test Index:  [15650 15651 15652 ... 19649 19650 19651]
```

Una vez realizada la validación, se tendría desarrollado al completo el modelo, mostrando como ejemplo la creación de un árbol de decisión, detallándose las diferentes partes de la implementación (ver Figura 56).

Figura 56

Ejemplo modelo árbol de decisión



6. Evaluación

En el presente apartado se procede al análisis de los resultados obtenidos con los modelos realizados, diferenciando entre los experimentos que se han llevado a cabo. Todas las métricas están calculadas sobre los datos del conjunto de test. Del mismo modo, el modelo seleccionado para extraer las métricas es el que mayor *accuracy* ha tenido de todos los realizados en el *cross-validation*.

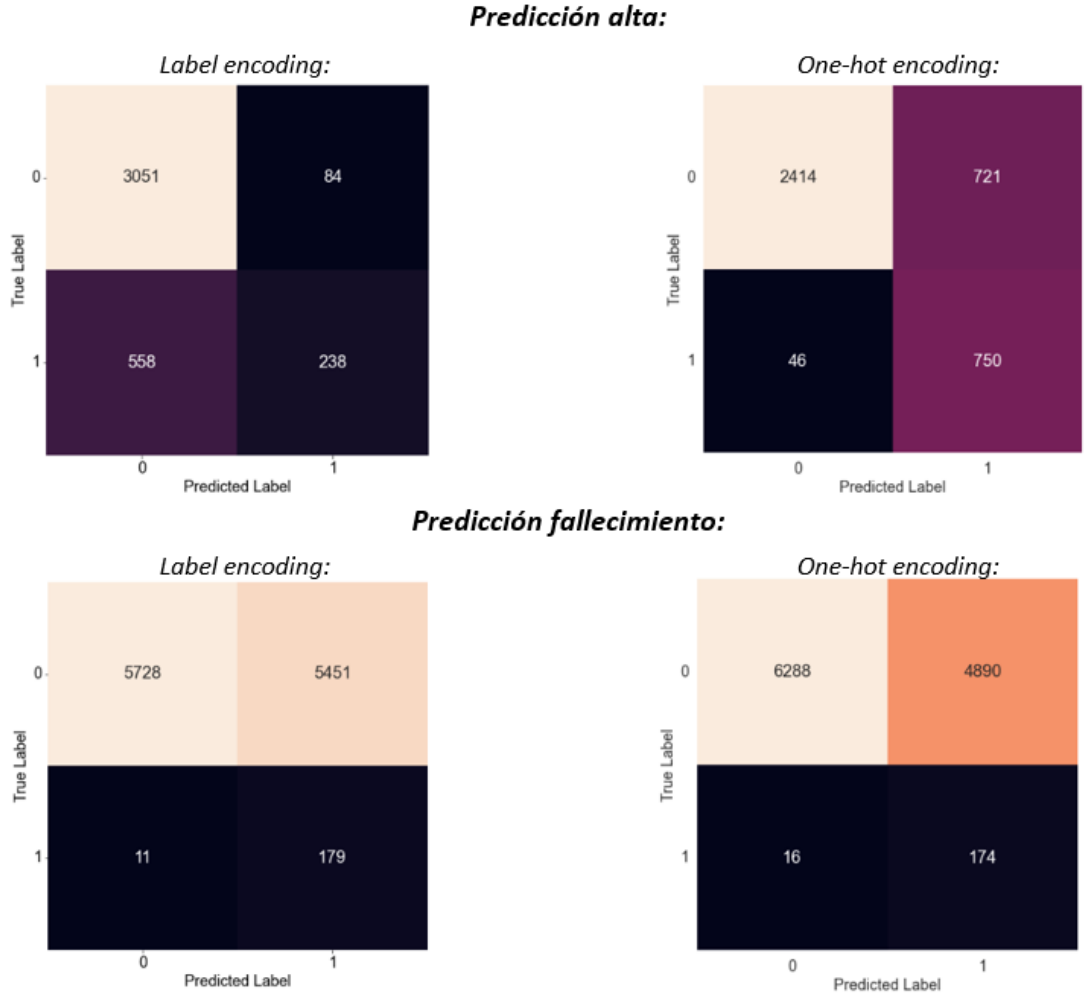
A continuación, se detallan y analizan las métricas obtenidas por los diferentes modelos predictivos, presentando los resultados para cada uno de los algoritmos de manera individual, finalizando con una comparativa entre todas las técnicas utilizadas.

- ***Naive Bayes***

En la Figura 57, se muestra las matrices de confusión de Naive Bayes, obteniendo el detalle de los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Figura 57

Matrices de confusión de Naive Bayes



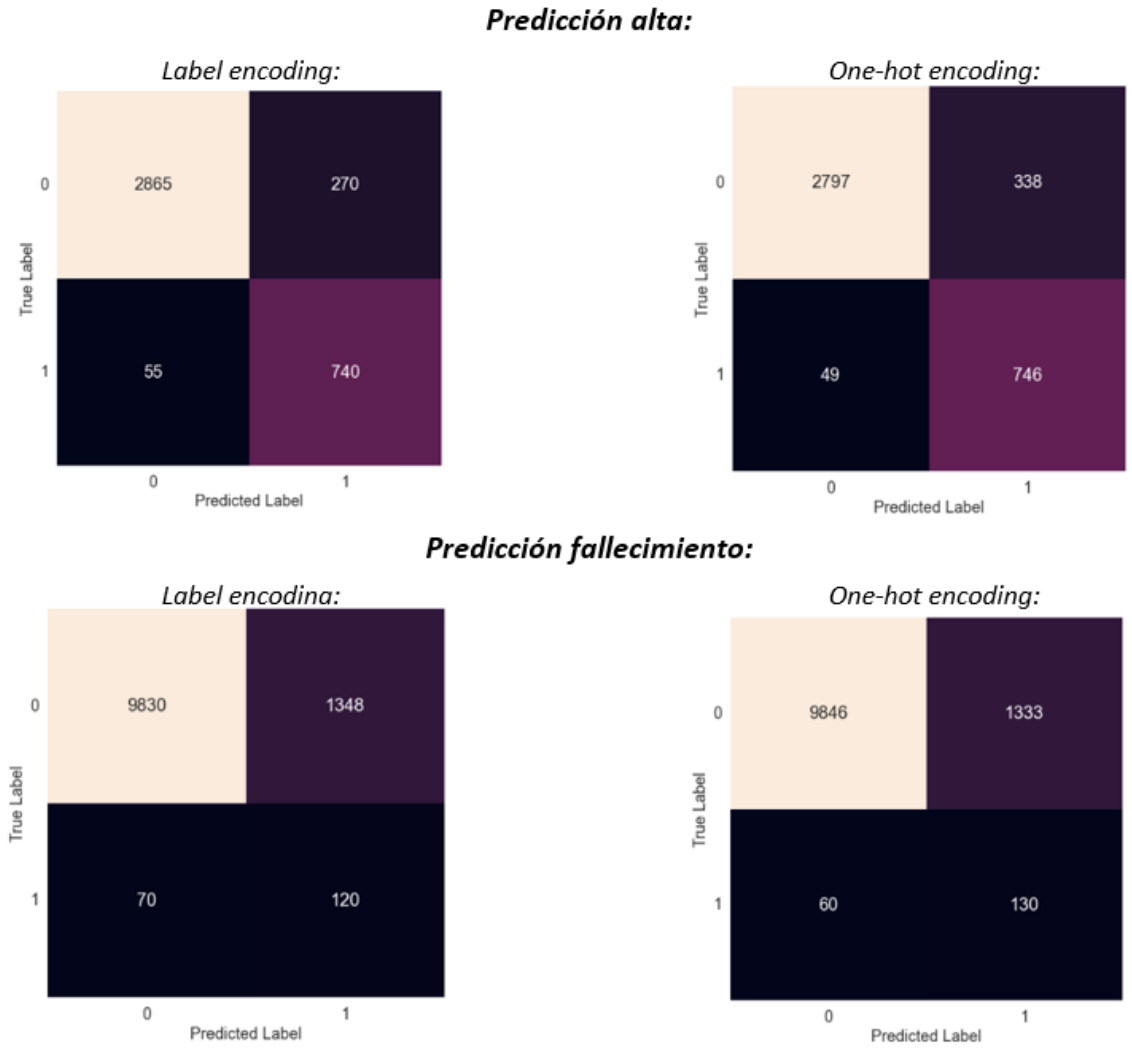
A la vista de estos resultados, se puede concluir que este modelo no es el más adecuado para los datos sobre los que se están realizando los experimentos, ya que no es capaz de efectuar ninguna clasificación efectiva, llegando a confundirse hasta en prácticamente un 50% de los casos.

- ***Árboles de decisión***

En la Figura 58, se muestran las matrices de confusión obtenidas tras analizar los resultados de los árboles de decisión. Viendo estas métricas, el éxito de estas predicciones varía según experimento, ya que los modelos que estiman el alta de una persona hospitalizada consiguen unos resultados correctos sin tener un gran volumen de fallos (FP y FN), mientras que las predicciones acerca del fallecimiento de un paciente obtienen una cantidad considerable de falsos positivos (FP).

Figura 58

Matrices de confusión de Árboles de decisión



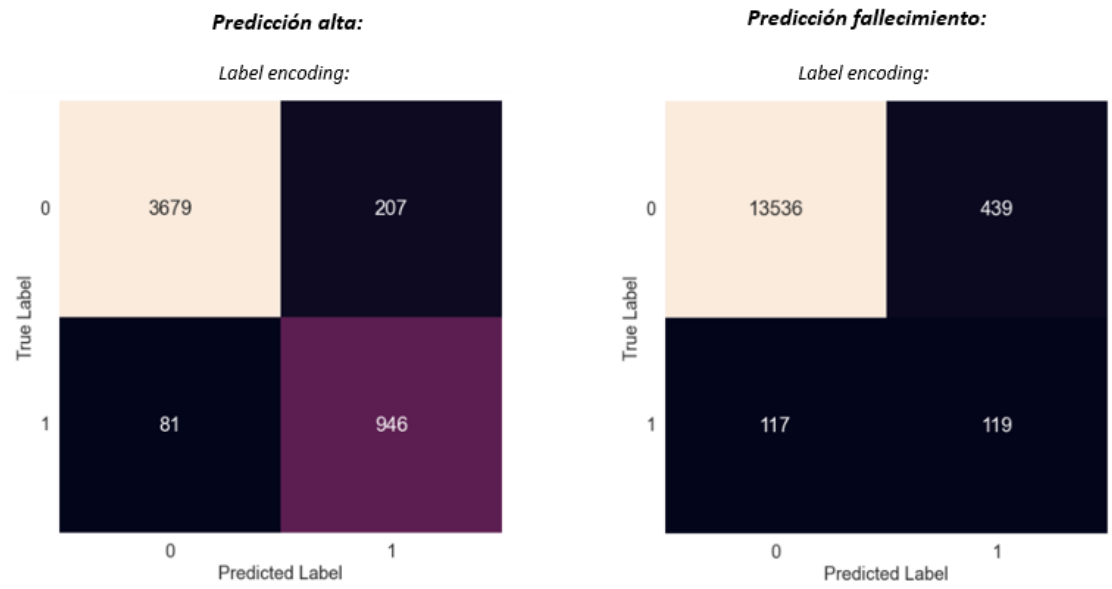
Esos errores al predecir que el paciente ha fallecido se deben principalmente a la poca cantidad de personas con alta por muerte que presentan los datos analizados, siendo incluso complicado de solventar con técnicas de desbalanceo.

- ***Random Forest:***

En la Figura 59, se puede observar las matrices de confusión obtenidas tras realizar las estimaciones con la técnica de Random Forest para los dos experimentos estudiados.

Figura 59

Matrices de confusión de Random Forest



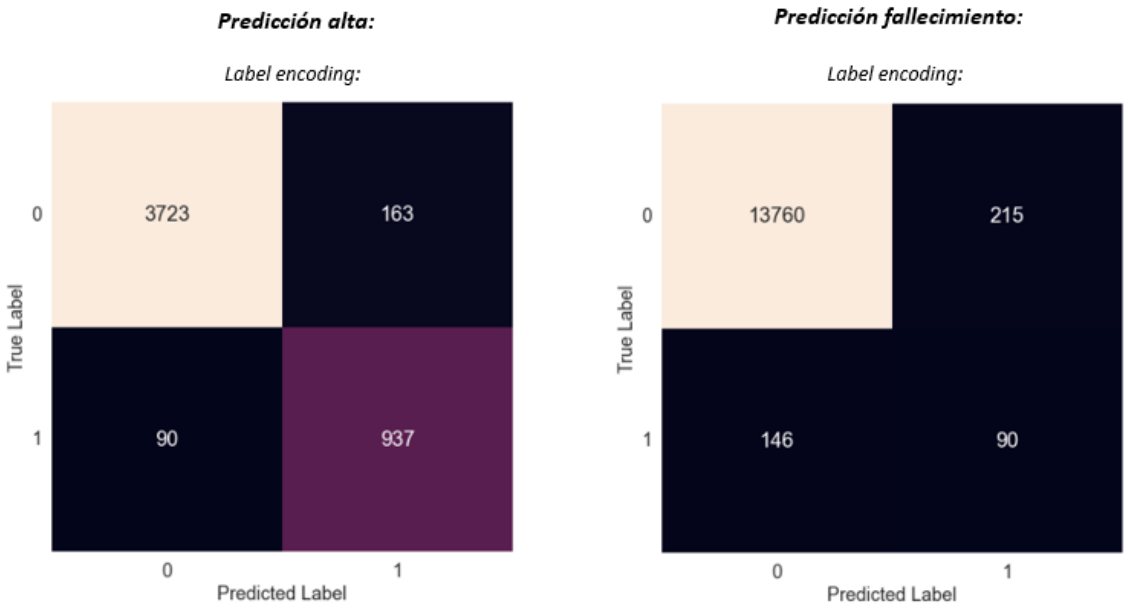
En este caso, los resultados obtenidos para la predicción del fallecimiento son mejores que los recogidos en los árboles de decisión, aunque a la hora de estimar los pacientes que han fallecido realmente falle de una manera considerable. Por otro lado, la matriz de confusión obtenida para la predicción del alta arroja unos resultados realmente buenos cometiendo pocos fallos.

- ***Gradient Boosting Decision Tree***

La Figura 60 muestra las matrices de confusión del modelo GBDT para las dos estimaciones propuestas. Al igual que en el modelo anterior, la capacidad de predicción para un paciente que realmente ha fallecido es bastante baja, siendo por la otra parte correcta para una persona que no ha recibido el alta por muerte, datos que reflejan el desbalanceo entre clases presente en el experimento. Por otra parte, se siguen observando unas buenas métricas para la estimación de alta de un paciente a partir de los datos de los traslados.

Figura 60

Matrices de confusión de Gradient Boosting Decision Tree



Los algoritmos basan sus decisiones en base a los datos de entrenamiento, pero algunas variables tienen mayor importancia que otras. A continuación, en la Figura 61, se muestra las variables que más importancia tienen en GBDT.

Figura 61

Importancia de variables GBDT

Predicción alta:

predictor	importancia
InHospitalTotalElapsedHoursNonCritics	0.429680
InHospitalNursingUnit9250ElapsedHours	0.099449
InHospitalNursingUnit9204ElapsedHours	0.072447
InHospitalSurgerySchedulingSurgeonCode	0.045644
InHospitalSurgeryInterventionRegisterElapsedMinutes	0.045542

Predicción fallecimiento:

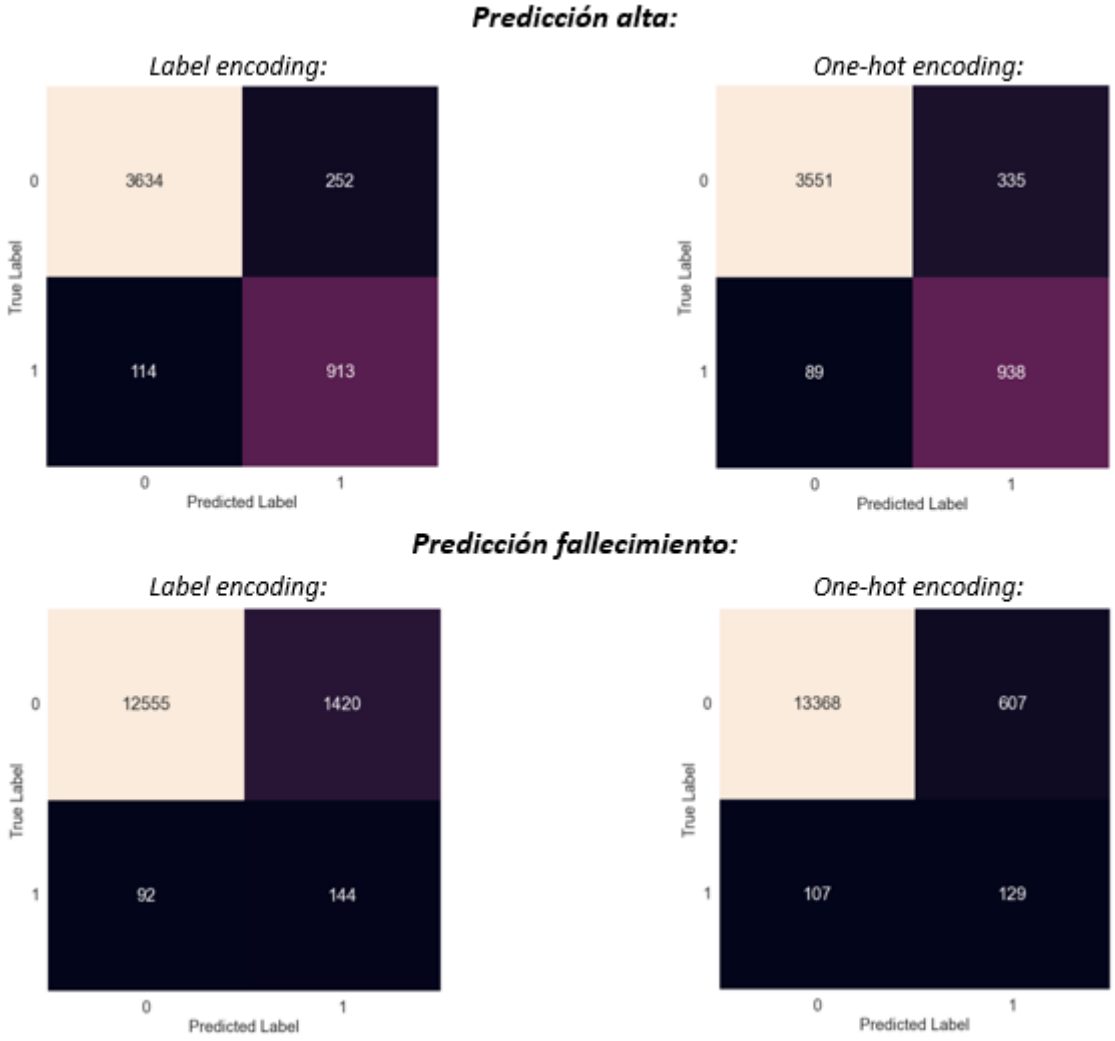
predictor	importancia
Age	2.553883e-01
InHospitalTimeSlotStart	1.623018e-01
InHospitalTotalElapsedHours	9.060006e-02
InErTotalElapsedHours	8.493904e-02
InHospitalNursingUnit	7.041548e-02

- ***Redes neuronales Perceptrón Multicapa (MLP)***

En la Figura 62, se muestran las matrices de confusión obtenidas a partir de las predicciones realizadas por las redes neuronales. Estas tablas reflejan un buen comportamiento en todas las predicciones, destacando especialmente en la estimación de alta por muerte, obteniendo el mejor resultado de todos los analizados.

Figura 62

Matrices de confusión red neuronal perceptrón multicapa



- ***Comparativa de modelos:***

En este apartado se establece una comparación entra las métricas recogidas en los diferentes modelos, diferenciando según la estimación realizada.

  - **Estimación de alta por fallecimiento del paciente:**

En la Tabla 6 se indican las métricas obtenidas para la estimación “alta por fallecimiento” con los diferentes algoritmos utilizados en su desarrollo. En ella pueden observarse los mejores resultados, destacados en color verde, y los peores, en color rojo.

Tabla 6

Métricas de los algoritmos para predicción de fallecimiento

Algoritmos	Codificación	Métricas				
		Accuracy	Recall	Precisión	Especificidad	AUC
Naive Bayes	Label	0.52	0.94	0.03	0.51	0.72
	One-hot	0.57	0.92	0.03	0.56	0.75
Árbol de Decisión	Label	0.88	0.63	0.08	0.88	0.87
	One-hot	0.87	0.68	0.09	0.88	0.90
Random Forest	Label	0.96	0.50	0.21	0.97	0.92
Gradient Boosting	Label	0.97	0.38	0.30	0.98	0.92
Red neuronal	Label	0.89	0.61	0.09	0.90	0.87
	One-hot	0.95	0.55	0.18	0.96	0.91

Al tratarse de un experimento en el que la variable a predecir sufre un gran desbalanceo entre sus clases, fenómeno este que afecta al comportamiento de los algoritmos, se han utilizado técnicas de oversampling y undersampling para poder reducir este efecto e intentar obtener mejores resultados.

Observando la Tabla 6, se puede concluir que entre los modelos seleccionados para el estudio los que mejor funcionamiento demuestran son la red neuronal empleando codificación one-hot y el algoritmo Random Forest. En ambos casos la métrica de precisión y especificidad, arroja un resultado que podría considerarse malo, pero se ve claramente afectado por el desbalanceo, ya que el número de verdaderos positivos es muy pequeño debido a los pocos elementos de esa clase.

- **Estimación del alta de un paciente**

En la Tabla 7 se presentan las métricas obtenidas para esta estimación con los diferentes algoritmos empleados. En ella se destacan, en color verde, los mejores resultados y, en color rojo, los peores.

Tabla 7

Métricas de los algoritmos para predicción del alta de un paciente

Algoritmos	Codificación	Métricas				
		Accuracy	Recall	Precision	Especificidad	AUC
Naive Bayes	Label	0.84	0.30	0.74	0.85	0.80
	One-hot	0.80	0.94	0.51	0.98	0.91
Árbol de Decisión	Label	0.92	0.93	0.73	0.98	0.96
	One-hot	0.90	0.94	0.69	0.89	0.96
Random Forest	Label	0.94	0.92	0.82	0.95	0.98
Gradient Boosting	Label	0.95	0.91	0.85	0.96	0.98
Red neuronal	Label	0.92	0.89	0.78	0.93	0.96
	One-hot	0.91	0.91	0.74	0.91	0.91

En ella se observa como el funcionamiento general de estos modelos, excluyendo los algoritmos de Naive Bayes, es bastante bueno. Consiguiendo para esta estimación unos resultados mejores que para el anterior experimento, echo que puede ser debido a que en este caso las clases a predecir no se encontraban tan desbalanceadas.

7. Conclusiones y trabajo futuro

7.1. Conclusiones

En el presente trabajo se ha extraído información de la mensajería HL7 de un hospital, creándose dos datasets; el primero de ellos incluye toda la información relacionada con cada episodio que genera un circuito de paciente que ha sido hospitalizado o ha tenido un episodio en urgencias. Se ha creado un segundo dataset, basado en el primero, pero añadiendo registros adicionales por cada traslado que han tenido los pacientes tanto en urgencias como en hospitalización. Con esa información, se han realizado unas predicciones con el fin de determinar si los datos recogidos, utilizados originalmente para labores de gestión en el centro hospitalario, pueden ser empleados también para predecir el comportamiento futuro que pueda tener un paciente mediante técnicas de aprendizaje automático y machine learning.

Se han establecido una serie de algoritmos y modelos predictivos para llevar a cabo las predicciones estimando, concretamente, el alta de un paciente (alta hospitalaria, alta voluntaria, alta por muerte, …) mediante la información de los traslados y prediciendo la posible muerte de una persona asistida en el hospital sabiendo los datos de los episodios hospitalarios y de urgencias.

Por último, se ha realizado un análisis de los resultados obtenidos por las predicciones para determinar la capacidad predictora de los datos recogidos. Del mismo modo, se han evaluado los diferentes modelos desarrollados, comparando el rendimiento dado por cada uno de ellos.

Tras efectuar todo lo anteriormente enunciado, se puede demostrar el uso de esta información para llevar a cabo estudios o implementaciones de modelos predictivos que permitan estimar diferentes variables, ya que los resultados obtenidos han sido correctos y las métricas analizadas han arrojado un buen funcionamiento de los algoritmos. Aunque, no hay que olvidar que en el caso de la predicción de alta por muerte los resultados son mejorables, por lo que sería necesario incorporar nuevas variables que tengan más capacidad predictiva y/o aumentar el dataset con más datos etiquetados con ‘muerte’.

Por otro lado, y más allá de las cuestiones relacionadas con el empleo de una u otra técnica, se destaca en este trabajo que, aunque el conjunto de datos cuenta con un gran porcentaje de valores ausentes y tiene un gran desbalanceo entre las clases, los métodos que se han empleado han demostrado que se comportan muy bien y que ofrecen un gran número de posibilidades. De igual manera, se extrae de la elaboración de este trabajo la importancia de hacer un buen procesamiento de los datos. El filtrado de filas o columnas poco interesantes, la imputación de los valores ausentes o la estandarización de los datos son cuestiones que hay que tener muy en cuenta durante el proceso de la creación de los modelos.

7.2. Líneas de trabajo futuro

Pese a que los objetivos pautados a lo largo de este TFM han sido alcanzados, se detallan, a continuación, una serie de propuestas de trabajo futuras relacionadas con la predicción sobre mensajería HL7 en el ámbito de gestión hospitalaria:

1. **Aumentar el conjunto de datos.** Los datos proporcionados para este trabajo han sido del año 2021 (26/03/2021 al 10/04/2022); se podrían solicitar datos de años anteriores al COVID-19 para evitar los sesgos que han podido crearse a raíz de la pandemia. Por otro lado, se recomienda incorporar datos de otros centros hospitalarios pudiendo así generar una mayor diversidad y veracidad de los datos.

2. **Obtención de nuevas variables.** Las variables calculadas para los datasets empleados están condicionadas con las predicciones que se quieren hacer. Por lo tanto, nuevas predicciones conllevarán nuevas variables a calcular. Se plantea la incorporación de nuevas variables que permitan mejorar el rendimiento de estas predicciones y desarrollar otras nuevas.

3. **Empleo de nuevos algoritmos.** Utilización de algoritmos que no han sido empleados en este estudio, analizando el funcionamiento de los mismos y determinando su rendimiento frente a los utilizados en el presente trabajo.

4. **Creación de nuevas predicciones.** Este trabajo se ha centrado en predicciones de traslado y de mortalidad (alta por muerte). Otras posibles predicciones a realizar pueden relacionarse con los tiempos de hospitalización, tiempos en unidades de enfermería, etc.

5. **Creación de software y prototipo.** Un fututo objetivo sería preparar un software que se conectara directamente con el Event Manager y de esta forma se pudieran realizar predicciones en tiempo real.

8. Glosario

1. CIC: Código de identificación corporativo de un paciente dentro de la organización. Es decir, el código que identifica de manera unívoca a ese paciente en el sistema de salud.
2. Circuito: Se define circuito como “el viaje” que realiza el paciente a través del hospital desde el ingreso hasta el alta. El presente trabajo incluye las siguientes casuísticas de circuito:
    - Urgencia
    - Hospitalización
    - Urgencia + hospitalización
3. Consulta ambulatoria de urgencia: Se trata del destino de los traslados que revisten menor gravedad en urgencia (Niveles de triaje de 3 a 5). Una vez determinado el nivel de gravedad los pacientes son atendidos en consulta, donde se realiza el diagnóstico del mismo y en la que se determina si recibe:
    - Alta
    - Ingreso en urgencia
    - Permanencia en boxes
4. Código CIE10 (Clasificación internacional de enfermedades). Es un catálogo de códigos que identifican las enfermedades.
5. Código único: Dentro de la terminología corporativa de Osakidetza se llama código único al dato que hace referencia de forma unívoca a una ubicación física que puede ser del tipo:
    - Local: Puede albergar a varios pacientes a la vez, como por ejemplo salas de espera.
    - Punto de atención: Sólo puede estar ocupado por un paciente, como por ejemplo boxes, camas, etc.
6. Episodio hospitalario: El episodio en el contexto del presente trabajo es el código que hace referencia a una asistencia hospitalaria o circuito desde el momento del ingreso, hasta el momento de alta. Todos los mensajes HL7 recibidos durante un episodio están vinculados con este código. En caso de recibir el alta y un nuevo ingreso, el episodio cambia y este es el motivo por el que para generar registros en formato plano que incluyan el circuito de urgencia y el circuito de hospitalización debemos aplicar reglas en el pre-procesador que los vincule. Esta vinculación se hace a través del motivo de alta de Urgencia (InErDischargeCode). Cuando un alta viene informada con este campo con el valor 10 significa que se trata de un alta por ingreso, por lo que se aplica la regla de combinar los episodios en un circuito de urgencia + hospitalización.

7. HIS: Sistema de información hospitalario en el que habitualmente se introducen funcionalides de historia clínica.

8. HL7 (Health Level Seven): es un conjunto de estándares para facilitar el intercambio electrónico de información clínica; que utiliza una notación formal del lenguaje unificado de modelado (Unified Modeling Language, UML) y un metalenguaje extensible de marcado con etiquetas (Extensible Markup Language, XML).

9. Mensaje HL7 versión 3: Mensaje en lenguaje de marcado XML que transmite un cambio en un episodio entre dos aplicaciones que se comunican bajo este protocolo. Por ejemplo, un traslado (ADT_A02) propaga información sobre el cambio de ubicación del paciente.

10. Motivo de alta: Valor catalogado de 1 a 10 que indica el motivo de alta. Se han incluido dos campos en los datasets que indican el motivo de alta para urgencia y hospitalización

11. Programación de intervención: Registro que parte de una solicitud de intervención que es agendada a la que cual se le asignan recursos físicos (bloque quirúrgico, quirófano, profesional, etc.). Se propaga a través del mensaje ITR_I03

12. Solicitud de intervención: Una solicitud de intervención es el registro relacionado con la solicitud de intervención quirúrgica que propaga el mensaje ITR_I01. Entre los datos que propaga se incluyen algunos datos diagnósticos como el código CIE10.

13. Registro de intervención: Registro que se informa una vez finalizada una intervención quirúrgica, y que se propaga a través del mensaje ITR_I02.

14. TimeStamp: En el presente trabajo se utiliza este tiempo para determinar con exactitud una marca de tiempo compuesta por fecha y hora, y que es fácilmente manejable, convertible y procesable mediante lenguajes de programación.

15. Triaje: Proceso a través del cual se mide el nivel de gravedad de un paciente que se realiza inmediatamente después del ingreso administrativo en urgencia. Arroja valores del 1 al 5 (de mayor a menor gravedad).

16. Unidad de enfermería: Conjunto de códigos únicos que componen una unidad asistencial funcional. En el caso de Osakidetza se identifican por un código entero de 4 dígitos que comienza por 9. Ejemplo: 9250 - Unidad de recepción de pacientes. Dentro del presente trabajo se realiza la división entre:

    - Unidades no críticas: La asistencia hospitalaria prestada en esta unidad no es crítica.

    - Unidades críticas: La asistencia hospitalaria prestada es crítica.

9. Bibliografía

Alekh, Sunder, S. &., & Gaur. (2015). *Building a DICOM/HL7 compliant Anonymizer Service conforming to HIPAA De-identification Regulations.* 10.13140/RG.2.1.2481.0967.

Baxt , W. (1992). Analysis of the clinical variables driving decision in an artificial neural network trained to identify the presence of myocardial infarction. Annals of emergency medicine, 21(12). *https://doi.org/10.1016/s0196-0644(05)80056-3*, 1439–1444.

Bishop, C. M. (1995). Neural networks for pattern recognition. *Oxford university press*.

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning. *New York: springer*, (Vol. 4, No. 4, p. 738).

Breiman, L. (1996). Bagging Predictors. Machine Learning, 24. 123-140.

Breiman, L. (2001). Random forests. Machine learning, vol. 45(1). 5-32.

Cooper, G. F., Aliferis, C. F., Ambrosino, R., Aronis, J., Buchanan, B. G., Caruana, R., . . . Spirtes, P. (1997). An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial intelligence in medicine, 9(2), 107–138 https://doi.org/10.1016/s0933-3657(96)00367-3*.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning. 273–297.

Dietterich, T. G., & et al. (2002). Ensemble learning. The handbook of brain theory and neural, 2(1). 110-125.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*, 103–130.

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. *Management Review (AMA FORUM), 70*, 35-36.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Hadzikadic, M. (1992). Automated design of diagnostic systems. Artificial Intelligence in Medicine. *https://doi.org/10.1016/0933-3657(92)90019-L*, 4(5), 329–342.

Islam, M. J., Wu, Q. J., Ahmadi, M., & Sid-Ahmed, M. A. (2007). Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers. *In 2007*

International Conference on Convergence Information Technology (ICCIT 2007). IEEE, 1541-1546.

John Lu, Z. Q. (2010). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. *Journal of the Royal Statistical Society Series A, 173, issue 3*, 693-694.

King, Z., Farrington, J., Utley, M., Kung, E., Elkhodair, S., Harris, S., . . . Crowe, S. (2022). *Machine Learning for Real-Time Aggregated Prediction of Hospital Admission for Emergency Patients.* EuropePMC.

Kline, J. A., Courtney, D. M., Kabrhel, C., Moore, C. L., Smithline, H. A., Plewa, M. C., . . . Nordenholz, K. (2008). Prospective multicenter evaluation of the pulmonary embolism rule-out criteria. *Journal of Thrombosis and Haemostasis, 6(5):772-780. https://doi.org/10.1111/j.1538-7836.2008.02944.x*.

Koren, Y. (2009). The Bellkor solution to the Netflix Grand Prize. Netflix prize documentation, vol. 81. 1–10.

Larrañaga P, Calvo B, Santana R, Bielza C, Galdiano J, Inza I, . . . Robles V. (2006). Machine learning in bioinformatics. *Brief Bioinform.; https://doi.org/10.1093/bib/bbk007. PMID: 16761367*, 86-112.

Leinweber, D. (2013). *Forbes*. Obtenido de https://www.forbes.com/sites/davidleinweber/2013/04/26/big-data-gets-bigger-now-google-trends-can-predict-the-market/

Maglogiannis, I., Karpouzis, K., Wallace, M., & Soldatos, J. (2007). Emerging Artificial Intelligence Applications in Computer Engineering - Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies. *IOS Press*.

Marewski, J. N., & Gigerenzer, G. (2012). Heuristic decision making in medicine. Dialogues in clinical neuroscience, 14(1), 77–89. *https://doi.org/10.31887/DCNS.2012.14.1/jmarewski*.

Miller, M. (1977). *Medical Diagnostic Models: A Bibliography by M. Clinton Miller, III ... [et Al.], volume 1.* University Microfilms.

Morales, E., & Escalante, H. J. (2012). Aprendizaje bayesiano. *Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla, México.*

Raita, Y., Goto, T., & Faridi, M. (2019). *Emergency department triage prediction of clinical outcomes using machine learning models.* Crit Care. 2019 Feb 22;23(1):64. doi: 10.1186/s13054-019-2351-7. PMID: 30795786; PMCID: PMC6387562.

Reggia, J. A. (1993). Neural computation in medicine. Artificial Intelligence in Medicine, 5(2), 143–157. *https://doi.org/10.1016/0933-3657(93)90014-T*.

Sahni, N., Simon, G., & Arora, R. (2018). *Development and Validation of Machine Learning Models for Prediction of 1-Year Mortality Utilizing Electronic Medical Record Data Available at the End of Hospitalization in Multicondition Patients: a Proof-of-Concept Study.* Journal of General Internal Medicine, 33(6), 921–928. doi:10.1007/s11606-018-4316-y.

Silver, N. (2008). *FiveThirtyEight*. Obtenido de https://projects.fivethirtyeight.com/2020-election-forecast/

Swain, M., & Kharrazi, H. (2015). *Feasibility of 30-day hospital readmission prediction modeling based on health information exchange data.* 84(12):1048-56. doi: 10.1016/j.ijmedinf.2015.09.003. Epub 2015 Sep 14. PMID: 26412010.

Wu, J., Grannis, S. J., Xu, H., & & Finnell, J. T. (2016). *A practical method for predicting frequent use of emergency department care using routinely available electronic registration data.* 16(1), 12. doi:10.1186/s12873-016-0076-3.

Zhu, K., Lou, Z., Zhou, J., Ballester, N., Kong, N., & Parikh, P. (2015). *Predicting 30-day Hospital Readmission with Publicly Available Administrative Database. A Conditional Logistic Regression Modeling Approach.* Methods Inf Med. 2015;54(6):560-7. doi: 10.3414/ME14-02-0017. Epub 2015 Nov 9. PMID: 26548400.

Anexos

Anexo. Artículo de investigación

Creación de datasets y modelos predictivos basados en mensajería HL7

Barrera Mayo, Joseba

Méndez Gutiérrez, Pablo

Vaquero Muñoz, Manuel

Universidad Internacional de la Rioja, Logroño (España)

Fecha 2022/09/21

Resumen

Este trabajo define la estrategia para generar datasets anonimizados a partir de mensajes transmitidos por aplicaciones médicas bajo el estándar de comunicaciones HL7. Se emplean técnicas de aprendizaje automático para predecir el tipo de alta, en el que se incluye alta por muerte a partir del circuito por el paciente durante su episodio. El enfoque se compone de los siguientes pasos: (i) Solicitud de los datos; (ii) anonimización; (iii) pre--procesado; (iv) creación de datasets; (v) creación de varios modelos de aprendizaje automático y ajuste de sus hiperparámetros y del umbral de probabilidad para las predicciones, y (vi) comparación de resultados entre los distintos modelos sobre los conjuntos creados para determinar la mejor solución. Los resultados muestran que una buena imputación de los valores ausentes y el ajuste del umbral de probabilidad son factores clave a la hora de mejorar el rendimiento de los clasificadores.

Palabras Clave

Anonimización de datos, aprendizaje automático, datos desbalanceados, mensajería HL7

I. Introducción

La gestión de los recursos hospitalarios es clave, sobre todo en momentos de alerta sanitaria como los que se han estado sufriendo debido a la pandemia originada por la Covid 19. Se calcula que el Servicio Vasco de Salud (Osakidetza), que da servicio a toda la comunidad autónoma además de algunos pueblos colindantes, ha atendido más de 16M de consultas en el año 2019, según su informe anual.

Toda esta actividad médica genera una gran cantidad de datos, que permiten tener en todo momento al paciente controlado y disponer de un historial clínico sobre él.

La mensajería HL7 es utilizada por miles de hospitales en todo el mundo como protocolo para el intercambio de información entre aplicaciones y sistemas del ámbito clínico. Este estándar permite comunicar datos entre equipos y software con sistemas que se encuentren en diferentes en la misma u otras organizaciones. Las normas HL7 actúan sobre una gran diversidad de datos clínicos como pueden ser el diagnóstico de un paciente, el ingreso en urgencias o el alta de este, pero también relacionados con la gestión del hospital como es el caso del cambio de cama de hospitalización o el transporte en ambulancia requerido por algunos pacientes.

El uso de esta tecnología permite comunicar aplicaciones de diferentes naturalezas; además, estandariza las comunicaciones y facilita la integración de sistemas en el ámbito sanitario.

Osakidetza dispone de una arquitectura propia de software a nivel centralizado, que recibe información de las aplicaciones corporativas y propagan esos datos mediante mensajería HL7 a las aplicaciones y sistemas suscriptores, quienes reciben en tiempo real esa información para poder realizar sus funciones.

Los mensajes enviados disponen de un timestamp y son procesados en tiempo real manteniendo un estricto orden correlativo, ya que cada uno representa una acción que debe ser ejecutada en orden el orden que ha tenido lugar.

En la actualidad, la inteligencia artificial está siendo utilizada en multitud de disciplinas entre ellas la medicina [13]. El aumento de la disponibilidad de datos transmitidos por aplicaciones clínicas ha posibilitado que su uso en el ámbito de la salud haya experimentado un gran crecimiento durante los últimos años, más si cabe con la pandemia del coronavirus, que ha sido un gran foco de estudio durante el último año.

La aplicación de la IA en la medicina puede permitir una mejora en la calidad de los sistemas sanitarios y un trato más optimo a los pacientes como, por ejemplo, el uso de nuevos métodos de detección de enfermedades empleando clasificación de imágenes o la utilización de diferentes sistemas robóticos para ayudar en la logística de un hospital.

Este trabajo se centra mayormente en la información administrativa transmitida mediante este protocolo y tiene como objetivo la creación de un procesador de datos que:

- Anonimice los mensajes HL7 eliminando los datos innecesarios y cifrando los datos necesarios para el correcto procesado de episodios en orden correlativo.

- Procesado de datos para generar diferentes datasets planos para crear una estructura de datos en que cada fila corresponda a un paciente a partir de su secuencia de mensajes HL7 obteniendo las variables necesarias para la creación de modelos predictivos basados en técnicas de aprendizaje automático.

Una vez obtenidos los datasets, se han creado y probado diferentes modelos para realizar predicciones en base el circuito realizado por el paciente dentro del hospital, capturando y generando variables con cada tipo de mensaje HL7 implementado en sus sistemas de información.

Para ello, se ha estudiado la documentación proporcionada por el hospital en base a los mensajes implementados y datos propagados, así como la arquitectura de sistemas con el objetivo de entender la forma de propagación y captura de la información. Se ha utilizado el histórico de mensajes de un año utilizado los relacionados con ingresos, traslados, triajes, intervenciones quirúrgicas, alta médica y altas. Posteriormente se ha generado una aplicación que procesa estos datos generando una estructura de clases que es exportada en formato JSON.

Posteriormente, esta estructura se ha procesado con lenguaje Python para normalizar los datos y aplicar técnicas de tratamiento de valores nulos. Con estos datos finales, se han creado modelos predictivos centrados en casos de uso muy específicos como la predicción del alta. Estos modelos se han evaluado obteniendo unos resultados y conclusiones que serán descritos en el apartado V.

II. Estado del Arte

En este capítulo se describen las técnicas de aprendizaje automático y profundo más utilizadas para la predicción en medicina en el ámbito que abarca el presente trabajo:

- Anonimización de datos en el ámbito de la salud
- Construcción de datasets y modelos predictivos que ayuden a optimizar la gestión hospitalaria

En el presente apartado se estudiarán algunos casos de uso desarrollados por otros autores y los modelos predictivos que se utilizarán a lo largo del trabajo.

La predicción básica se ha utilizado en diversos campos. Por ejemplo, Nate Silver, fundador y editor jefe de FiveThirtyEight, es un estadístico que utiliza modelos de predicción con el fin de predecir las elecciones estadounidenses con un alto grado de éxito gracias a la recopilación y el análisis de datos políticos para dar el posible resultado de las próximas elecciones [20].

Otro ejemplo es el de Google, que descubre las tendencias y las fluctuaciones del mercado bursátil [12] al encontrar la correlación entre las búsquedas en Internet del nombre de una empresa y su volumen de negocio, aunque no pudo predecir su precio en la bolsa.

En el campo de la medicina, en los años 60, se empezaron a utilizar sistemas de ayuda a la decisión de diagnósticos con aplicaciones bayesianas. Se propusieron varios métodos para eliminar el efecto de los resultados redundantes [15].

En la década de los 90, varios trabajos analizaron las técnicas de aprendizaje automático supervisadas y no supervisadas [7] aplicadas a la medicina. Un ejemplo de técnica supervisada es la red neuronal, formada por "elementos de procesamiento activo cuyas interacciones locales a lo largo del tiempo conducen al comportamiento global del modelo" [18]. En [2], Baxt utiliza las redes neuronales para identificar patrones y relaciones complejas que son difíciles de encontrar por un humano. La aplicación de redes neuronales para identificar el infarto agudo de miocardio en pacientes que acuden a urgencias cuando tienen dolor torácico anterior, ha demostrado ser más precisa que los médicos.

La heurística también se ha utilizado como análisis predictivo, ayudando a los médicos a tomar mejores decisiones. Con la heurística, la estrategia consiste en ignorar parte de la información y centrarse en un par de elementos clave para participar en la predicción. Marewksi y Gigerenzer [14] explican cuándo la heurística puede superar a otros métodos que utilizan mucha más información. El método consiste en hacer algunas preguntas de sí o no, como simplemente buscar una anomalía en el electrocardiograma del paciente, o por ejemplo si el paciente se queja de dolor en el pecho. Dentro de la misma línea de investigación, se creó una regla de decisión clínica muy sensible para decidir si se debe realizar o no una tomografía computarizada (TC) a los pacientes con traumatismos craneoencefálicos leves, debido a que sólo un pequeño porcentaje de pacientes empeora y requiere intervención médica. En ese caso, es necesario realizar un diagnóstico precoz del hematoma intracraneal mediante TC y la posterior intervención quirúrgica.

En [11] se utilizaron reglas de decisión para decidir si debía realizarse una prueba de embolia pulmonar (EP) en pacientes con factores de riesgo bajos, ya que los expertos sugieren que la EP sigue pasándose por alto en un alto porcentaje.

Otro uso popular del análisis predictivo es el desarrollo de métodos de aprendizaje automático para predecir la mortalidad en pacientes diagnosticados inicialmente de neumonía. Esto es útil para dar al médico una idea de si el paciente es apto para irse a casa o debe permanecer en el hospital para recibir más cuidados intensivos [5].

A continuación, se detallan algunos casos de uso en el ámbito de la inteligencia artificial en entornos de salud para la creación y validación de modelos predictivos, comenzando con la anonimización de datos. Se entiende caso de uso como un trabajo con una orientación vertical a la resolución de un problema concreto en un contexto concreto. Con el presente apartado se pretende detallar de qué datos parten, qué algoritmos utilizan y que resultados obtienen y en base a qué métricas.

Anonimización de datos

El trabajo de anonimización de datos estudiado, [1] se centra en los protocolos DICOM (digital imaging and communications in medicine) y HL7 (health level seven), estándares de comunicaciones para la transferencia de datos de imágenes y texto respectivamente entre aplicaciones en el ámbito sanitario basado en el estándar HIPAA (equivalente estadounidense al europeo GDPR) identifica los datos que deben ser cifrados o eliminados en el ámbito de salud. En el marco de los datos de texto incluido en protocolo HL7 habla sobre la pseudo-anonimización y la anonimización de la información detallando que la primera es aquella que permite asignar un pseudónimo a cada paciente con el objetivo de poder identificarlos a través de este sin que se conozca su identidad.

Obtención de variables de mensajes HL7 para la predicción de readmisión a 30 días

La readmisión hospitalaria no planificada representa grandes costes al sector sanitario. Los autores [21] describen cómo han identificado diferentes modelos de riesgo de reingreso hospitalario (Readmission Risk Prediction Models) y han estudiado las variables que se pueden extraer del conjunto de mensajería HL7 para alimentar estos modelos.

El estudio parte de trabajos publicados antes de 2013, extrayendo las variables utilizadas para la generación de los modelos y las buscan en el contenido de mensajes HL7. Identifican 297 variables predictivas tras la revisión de 32 artículos.

El trabajo consigue obtener el total de las variables extrayéndolas de los segmentos:

- DG1 (Diagnóstico): 89.2%
- PID (Identificación): 13.9%
- OBX (Observación): 9.1%

La publicación concluye que se pueden extraer datos mediante el procesado de mensajería HL7 siempre y cuando la integridad de estos datos supere ciertos umbrales.

Por otro lado, el trabajo Predicting 30-day Hospital Readmission with Publicly Available Administrative Database [23]. A Conditional Logistic Regression Modeling Approach utiliza "Condicional Logistic Regression" (CLR) y validación cruzada. El trabajo mejora resultados en comparación con algoritmos como "regresión logística directa", "regresión logística paso a paso", "random forests", "support vector machine".

El trabajo concluye que los modelos CLR puede ayudar a desarrollar a futuro modelos de predicción para identificar el riesgo de reingreso y subraya la importancia de la recopilación de datos sobre otros marcadores y su estructuración en bases de datos para poder desarrollar más estudios.

Predicción de uso frecuente de urgencias

El trabajo [22] desarrolla un modelo basado en regresiones multivariable que permite clasificar al paciente de urgencias como "frecuente" cuando tiene más de 4 visitas anuales. Este tipo de pacientes representa entre el 4.5% y el 8%.

Utiliza variables demográficas como edad, sexo. También utiliza el número de visitas durante un periodo (2018), así como la variable "Chief Complaint", que determina la categoría de las de la dolencia en base a un catálogo determinado. Otra de las variables que utiliza es la distancia que calcula a través del código postal de origen del paciente y código postal del centro

Incluye referencias a datos de segmentos concretos del segmento Patient Visit (PV1) y Patitent Identification (PID). Optan por excluir registros con datos faltantes y por la agrupación de episodios pertenecientes a un mismo paciente asignando un identificador de global único que les ayuda a emparejar todas las visitas de un paciente en diferentes bases de datos.

Determina que es técnicamente factible utilizar datos de registro para predecir este comportamiento.

Predicciones en triaje

En el estudio [17] Utilizan LASSO regression (least absolute shrinkage and selection operator), random forest, gradient boosted decision tree, y redes neuronales para priorizar pacientes en el momento del triaje a partir de datos demográficos, información de triaje, y datos de morbilidad asociada, con el objetivo de saber si van a necesitar acudir a una unidad de atención crítica (en la que se incluyen los casos de muerte intra-hospitalaria). El objetivo del trabajo es ayudar a asignar eficientemente los recursos hospitalarios, y priorizar pacientes de alto riesgo.

Obtienen resultados clasificando pacientes que necesitarán cuidados intensivos e ingreso hospitalario, obteniendo resultados AUC de 0.86 utilizando redes neuronales para cuidados intensivos y AUC de 0.82 para hospitalización con técnicas de machine learning. En el caso de hospitalización, obtienen peores resultados con redes neuronales.

Predicción de mortalidad a un año

El estudio [19] se centra en realizar una prueba de concepto para determinar si es posible predecir el riesgo de mortalidad a un año en el momento del alta hospitalaria a partir de variables demográficas, fisiológicas, biomédicas y de comorbilidad.

Realizan diferentes pruebas para completar los datos faltantes basadas en estrategia de vecinos y cálculo de mediana obteniendo los mismos resultados.

Aunque el trabajo no parte directamente de datos extraídos de mensajería HL7, sugiere que las variables utilizadas podrían extraerse de este estándar.

Utiliza Random Forests y regresiones logísticas obteniendo concluyendo que el uso de los primeros ayuda a trabajar con datasets con mucho ruido concluyendo además que las variables fisiológicas y bioquímicas son las más predictivas.

Predicción de ingreso en urgencias a corto plazo

El estudio Machine Learning for Real-Time Aggregated Prediction of Hospital Admission for Emergency Patients [10], utiliza técnicas de aprendizaje automático para predecir los ingresos a corto plazo.

Parten de una base de datosgenerada a partir de datos propagados mediante mensajería HL7 que es almacenada en una base de datos relacional PostgresSQL, La base de datos incluye el registro completo del paciente, incluidas las observaciones, resultados de las patologías, la ubicación de los pacientes, las solicitudes de consulta y un resumen del historial de visitas anteriores.

Utiliza clasificadores XGBoost y sugirie que se puede mejorar los servicios hospitalarios si utilizando esta técnica que ofrece buenos resultados.

III. Objetivos y Metodología

Objetivo

El objetivo de este estudio es realizar predicciones sobre datos extraídos a partir de mensajes HL7, utilizados estos para la gestión de un hospital. Se establece como prueba de concepto la realización de dos predicciones:

a. Predecir la mortalidad de un paciente a partir de los episodios completos en urgencias y hospitalización que se han recogido durante un tiempo determinado.

b. Prever el alta de un paciente en base a los datos obtenidos en el conjunto de traslados de los usuarios ingresados en el hospital.

De tal forma, que el trabajo pretende demostrar que esta información que antes tenía un uso mayormente administrativo puede ser empleado para más cuestiones, en este caso la predicción del comportamiento del paciente.

Metodología

Con el fin de alcanzar el objetivo previamente propuesto, se establecen una serie de tareas que se deben realizar, las cuales se detallan a continuación:

1. **Solicitud de datos:** Cumplimiento de las exigencias marcadas por la entidad hospitalaria suministradora de los

datos.

2. **Anonimización de los datos:** Desarrollo de un software que permita anonimizar ciertos datos personales y eliminar otros que faciliten la identificación de un paciente.
3. **Pre-procesado de los datos:** Desarrollo de una herramienta que mapea en una estructura de clases todos los mensajes analizados.
4. **Creación de datasets:** Desarrollo de las tablas de datos, a partir del software utilizado para procesar la información. Generando dos alternativas:
   a. Tabla con la información de los episodios completos en urgencias y hospitalización de los pacientes.
   b. Tabla que almacena el histórico de traslados de los pacientes.
5. **Creación de modelos predictivos:** Tratamiento previo sobre los datos para poder realizar estimaciones sobre ellos y realización de diferentes algoritmos para efectuar las predicciones estimadas en el trabajo.
6. **Obtención de resultados:** Analizar el rendimiento de los modelos predictivos realizados y comparar las métricas obtenidas en cada uno de ellos.

IV. Contribución

La revisión de la bibliografía revela que existen muchos trabajos relacionados con los algoritmos predictivos en el ámbito de la medicina. Tal y como se describe en los apartados anteriores, existen relaciones con los tres aspectos que aborda el proyecto:

Anonimización previa de los datos

Se parte del trabajo de [1] para identificar técnicas de anonimización de datos HL7 basados en la normativa HEPAA. En el caso del presente trabajo se desarrollarán técnicas para el procesado de la mensajería basados en tecnologías .NET que nos facilitarán la generación de grandes estructuras de datos y no nos ayudarán a eliminar en un primer paso los datos identificativos más relevantes procesando grandes volúmenes de información, para poder generar datasets totalmente anonimizados en un segundo paso, cumpliendo con los estándares citados en el artículo.

Generación de datasets

Los trabajos analizados que utilizan mensajería HL7 parten de información fuertemente relacionada con segmentos de información que albergan datos clínicos: DG1 (Diagnóstico) y OBX (Observaciones). El presente trabajo trata de aportar en este ámbito un enfoque diferente, como por ejemplo el uso de información relacionada con los movimientos del paciente a través del hospital, para generar datasets con información referente a la estancia en ciertas unidades de enfermería, así como información relacionada con posibles reingresos, posibilidad de muerte, o necesidades de recursos específicos. El trabajo se centra por lo tanto en generar predicciones en base a otro de información contenida en los segmentos ADT (Administrativo) e ITR (intervenciones quirúrgicas) que pudiera aportar variables relevantes para aplicar a ciertos casos de uso.

Por otro lado, cabe destacar que otros trabajos parten de bases de datos estructuradas en forma de datasets planos. El presente estudio no implementa modelos basados en datasets ya creados, sino que tiene un importante componente en el ámbito de la minería de datos, partiendo de una secuencia de mensajes HL7 e implementando un procesador que mapea y estructura esta secuencia en datasets que luego serán utilizados para entrenar y testear modelos.

Como se ha comentado, cada mensaje recibido está relacionado con una acción sobre un episodio. Una vez anonimizado el mensaje se va construyendo cada registro del dataset en base al procesamiento secuencial de todos los mensajes. El trabajo aporta el desarrollo de un software capaz de leer una cola de mensajes y generar una estructura de datos en memoria que después es exportada a JSON en base a un procesado secuencial de los mismos en memoria teniendo en cuenta los siguientes aspectos:

- Se han filtrado los pacientes y, se han eliminado aquellos que no tengan al menos un ingreso en urgencias y un ingreso en hospitalización.
- Se ha subdividido la lista de mensajes, cada vez que se detecta un ingreso, ya sea por urgencia o por hospitalización. De esta forma se puede considerar como nuevo paciente al grupo de mensajes comprendidos entre un ingreso y un alta.
- La tercera decisión consistió en fusionar aquellos pacientes cuya alta era del tipo 10 (alta por ingreso) y el ingreso fuera de hospitalización (ver Figura 26).

Una vez procesados los registros se ha generado un dataset plano. A continuación, se muestran los más importantes:

- Datos del paciente:
  - Edad
  - Sexo
  - SI dispone de alergias
  - Código de identificación cifrado
- Datos administrativos en urgencia:
  - Número de triajes
  - Número de traslados
  - Número de horas en unidad de urgencia
  - Franja horaria en urgencias
  - Motivo de alta
- Datos administrativos en hospitalización
  - Número de traslados
  - Tiempo en hospitalización
  - Slot horairo de ingreso
  - Slot horario de alta
  - Código de ingreso en hospitalización
  - Es crítica la unidad de hospitalización
  - Unidad de hospitalización
  - Motivo de alta
  - Solicitud de intervención
  - Tiene cambio de quirófano
  - Prioridad de solicitud de intervención
  - Código de sección de hospitalización
  - Código de procedimiento de intervención
  - Tiene intervención de alta en urgencia
  - Código del procedimiento de intervención
- Tiempos en unidades. Se ha creado una columna con el tiempo en cada unidad de enfermería del paciente
- Recorrido: Se ha guardado en una columna el recorrido por

las diferentes unidades de enfermería

Para almacenar los datos se ha empleado la tecnología de Microsoft Entity Framework, ya que como cualquier ORM, permite acceder a una base de datos utilizando clases que representan cada una de las entidades de ésta, pudiendo realizar cualquier operación sobre los datos simplemente llamando a métodos de estas clases.

Se ha conseguido procesar los 2.2M de registros dependiendo del PC (I7 9G 16GB RAM) en 30 minutos aproximadamente.

Modelos

La mayoría de los estudios o experimentos que utilizan esta tipología de datos centra más su atención en el algoritmo a elegir, que en el resto de las acciones que hay que llevar a cabo. En el presente trabajo, además de realizar varios modelos predictivos, se destaca las técnicas realizadas para tratar un conjunto de datos que cuenta con un gran porcentaje de valores ausentes y tiene un gran desbalanceo entre las clases a predecir.

Se considera una aportación el desarrollo de un software que sea capaz de llevar a cabo esas predicciones de manera adecuada, ya que puede suponer el primer paso para el desarrollo de una herramienta que facilite la gestión eficiente de los recursos en el ámbito sanitario.

A continuación, se detalla el proceso efectuado para realizar los modelos de aprendizaje automático e inteligencia artificial que han sido aplicados en el trabajo. Del mismo modo, se especifican las estrategias que se han seguido para configurar estos algoritmos de manera apropiada, explicando en qué consisten estas técnicas y qué proporcionan a los algoritmos.

Las clases que se predicen en estos modelos están claramente desbalanceadas, las altas por muerte son significativamente menores al resto de casos que suponen el fin de un episodio hospitalario. Del mismo modo, el hecho de ser dado de alta dentro de un conjunto de traslados es inferior, ya que normalmente cuando se es hospitalizado se recorren una serie de unidades antes de finalizar la estancia.

Para tratar estos datos desbalanceados se ha utilizado el método de oversampling SMOTE, mediante el cual se ha aumentado la clase minoritaria, permitiendo crear dos variables equilibradas. Esta generación de nuevos registros se basa en el algoritmo KNN (k-Nearest Neighbour).

Naive Bayes

Algoritmo de clasificación [8] basado en el Teorema de Bayes [16], se trata de un modelo simple y rápido, que no requiere de ningún tipo de parámetro para ajustar su comportamiento.

Para estimar el rendimiento del algoritmo se realiza validación cruzada; esta técnica consiste en dividir los datos en k particiones, en este caso 5, creándose un modelo que utiliza 4 particiones para entrenar y una restante para evaluar; se realiza este proceso 5 veces desarrollándose las diferentes combinaciones posibles, actuando finalmente todas las particiones una vez como validación. Al utilizar unos datos desbalanceados, se requiere el uso de la validación cruzada Stratified, que permite mantener la distribución de clases en cada una de las particiones seleccionadas.

En cada una de las iteraciones que realiza la cross-validation se aplica sobre los datos de entrenamiento la técnica SMOTE con el fin de disminuir los problemas que pueda tener el algoritmo con las desigualdades entre clases a predecir, quedando el modelo implementado.

Árboles de decisión

Algoritmo predictivo [9] que, a partir de un conjunto de reglas, permite dividir el espacio de las variables predictoras agrupando observaciones con valores similares para la variable a estimar.

Con el fin de obtener la configuración del árbol que mejor encaje con los datos de estudio, se evalúa el rendimiento de este para diferentes profundidades, midiéndose en cada una el accuracy, recall y precisión, y seleccionando aquella que mejor encaja.

Finalmente, se realiza la implementación del árbol con esa profundidad calculada y utilizando, de nuevo, validación cruzada y SMOTE.

Random Forest

Modelo predictivo [4], que consiste en la combinación de árboles de decisión, basándose en los principios de bagging [3] y en la selección de variables de forma aleatoria para cada uno de los árboles.

Para obtener la mejor configuración de este algoritmo con los datos de los que se dispone, se realiza una búsqueda de hiperparámetros mediante una rejilla aleatoria, que consiste en la definición de una serie de valores de ajuste y el empleo de ellos de manera aleatoria en el modelo, seleccionando el que mejor resultado arroje.

Gradient Boosting Decision Tree:

Algoritmo que se basa en la combinación de árboles de decisión de forma secuencial, de manera que cada nuevo árbol incorporado utiliza la información del anterior y trata de suplir los errores que ha podido cometer.

Al igual que para el modelo de random forest, para ajustar los hiperparámetros se utiliza una rejilla aleatoria. De ello resulta un código de instalación para este modelo muy similar al anterior cambiando, únicamente, los valores de la rejilla y el modelo a invocar. Para evitar problemas de sobreajuste [6] podemos ajustar un parámetro llamado learning rate o shrinkage (tasa de aprendizaje) que gradúa cuánto se va corrigiendo el error en cada iteración. Hará falta un número mayor de iteraciones (árboles) cuando se introduce un valor de shrinkage muy pequeño, mientras que para tasas moderadas el número de iteraciones necesarias es inferior.

Redes neuronales Perceptrón Multicapa (MLP):

Red neuronal compuesta por una capa de entrada, otra de salida y n capas ocultas entre ambas. Entrenar una red neuronal supone encontrar todos los valores de los pesos y bias de las neuronas que componen las diferentes capas.

La implementación conlleva muchos parámetros a decidir, como son:

- Número de capas.
- Cantidad de neuronas en cada capa.
- Funciones de activación de cada capa.
- Funciones de perdida y métricas durante la compilación.
- Numero de época y tamaño del bath en el entrenamiento.

El número de neuronas en la capa de entrada se corresponde con la cantidad de variables predictoras; y las neuronas en la capa de salida, con las variables a predecir, en este caso una.

Como función de activación en las capas ocultas se emplea

ReLU, ya que, computacionalmente, es la más eficiente y la no existencia de saturación en esta permite un entrenamiento más ágil. Para la capa de salida, se emplea una función de saturación sigmoide pues mapea los valores finales entre 0 y 1 (resultado binario). Al tratarse de un problema de clasificación binaria durante la compilación se utiliza una función de perdida binary_crossentropy y como métrica el accuracy.

Por otro lado, el resto de los parámetros se introducen tras probar diferentes configuraciones y comprobar las que mejor resultado dan al problema. Además, se ha empleado la técnica de early stopping que evita el sobre ajuste de la red neuronal a los datos de entrenamiento, parando la fase de aprendizaje de la red neuronal cuando las métricas sobre los datos de validación empiezan a saturar y dejan de arrojar mejores valores que las anteriores fases (epochs).

V. Evaluación y Resultados

En este estudio se han llevado a cabo dos predicciones con el fin de demostrar que los datos empleados pueden ser utilizados en otros casos que no sean los puramente administrativos. Para analizar el comportamiento y el rendimiento de las estimaciones establecidas se han efectuado las siguientes métricas:

- **Accuracy:** Especifica el porcentaje de predicciones verdaderas, es decir, aquellas que el modelo ha acertado.
- **Especificidad:** Mide los casos negativos que el algoritmo ha clasificado correctamente.
- **Recall:** Mide la proporción de casos positivos que fueron correctamente identificadas por el algoritmo.
- **Matriz de confusión:** Matriz en forma de tabla donde cada columna representa el número de predicciones de cada clase, mientras que cada fila muestra el número real de instancias de cada clase.
- **Curva ROC:** Indica la ratio de positivos reales versus la ratio de positivos falsos en distintos umbrales de clasificación.
- **AUC:** Se trata de un simple integral de la superficie que se encuentra debajo de la curva ROC. Un modelo perfecto tendría un AUC de 1, mientras que un modelo pésimo tendría un valor de 0.

Evaluación 1

En este punto se va a evaluar la predicción de alta por fallecimiento de un paciente a partir de los episodios hospitalarios o de urgencias. Hay que considerar que la información utilizada para realizar esta estimación está altamente desbalanceada, teniendo 948 registros para la clase ‘muerte’ y 55893 para la ‘no muerte’.

Algoritmos	Codificación	Métricas				
		Accuracy	Recall	Precisión	Especificidad	AUC
Naive Bayes	Label	0.52	0.94	0.03	0.51	0.72
	One-hot	0.57	0.92	0.03	0.56	0.75
Árbol de Decisión	Label	0.88	0.63	0.08	0.88	0.87
	One-hot	0.87	0.68	0.09	0.88	0.90
Random Forest	Label	0.96	0.50	0.21	0.97	0.92
Gradient Boosting	Label	0.97	0.38	0.30	0.98	0.92
Red neuronal	Label	0.89	0.61	0.09	0.90	0.87
	One-hot	0.95	0.55	0.18	0.96	0.91

Tabla 1. Métricas de los algoritmos para predicción de fallecimiento.

En la Tabla 1, se indican las métricas obtenidas para la estimación “alta por fallecimiento” con los diferentes algoritmos utilizados en su desarrollo. En ella pueden observarse los mejores resultados, destacados en color verde, y los peores, en color rojo.

Observando la Tabla 1, se puede concluir que entre los modelos seleccionados para el estudio los que mejor funcionamiento demuestran son la red neuronal empleando codificación one-hot y el algoritmo random Forest. En ambos casos la métrica Recall, arroja un resultado que podría considerarse malo, pero se ve claramente afectado por la desigualdad de clases, ya que el número de verdaderos positivos es muy pequeño debido a los pocos elementos de esa clase.

Como se ha especificado, el experimento sufre un gran desbalanceo entre sus categorías (‘muerte’, ’no muerte’), fenómeno este que afecta al comportamiento de los algoritmos, para intentar reducir este efecto se han utilizado técnicas de oversampling [SMOTE] y undersampling [Random UnderSampler], permitiendo mejorar los resultados y obteniendo predicciones como la mostrada en la Figura 1.

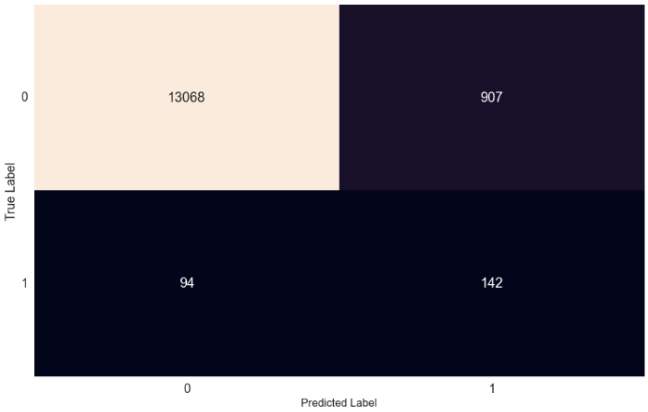


Fig. 1. Matriz de confusión para red neuronal codificación one-hot, donde la categoría 1 representa alta por muerte y la 0 ‘no muerte’.

Evaluación 2

En este otro punto, se va a evaluar la predicción de alta de un paciente a partir de los datos del conjunto de traslados hospitalarios, es decir se va a determinar si el próximo paso de una persona hospitalizada es un alta. Hay que tener en cuenta que esa alta puede ser de diferente naturaleza, por ejemplo, alta de hospitalización, alta voluntaria, alta por muerte, etc.

El alto número de unidades de enfermería, derivado del gran tamaño de la organización hace muy difícil predecir cuál va a ser el siguiente traslado. El motivo es que existe un conjunto muy grande de posibilidades y cada registro del dataset dispone de muchas variables con valores nulos (unidades por las que dicho paciente no pasa). Por este motivo se ha optado por predecir si el siguiente movimiento va a ser un alta. Éste alta libera recursos en el hospital, por lo que puede resultar una predicción objetivamente útil. Quizá en organizaciones más pequeñas y con conjuntos de datos más amplios se podría llegar a desarrollar el modelo que permita predecir el siguiente movimiento, ya que existirían menos valores faltantes.

En la Tabla 2, se presentan las métricas obtenidas para esta estimación con los diferentes algoritmos empleados. En ella se destacan, en color verde, los mejores resultados y, en color rojo, los peores.

Algoritmos	Codificación	Métricas				
		Accuracy	Recall	Precision	Especificidad	AUC
Naive Bayes	Label	0.84	0.30	0.74	0.85	0.80
	One-hot	0.80	0.94	0.51	0.98	0.91
Árbol de Decisión	Label	0.92	0.93	0.73	0.98	0.96
	One-hot	0.90	0.94	0.69	0.89	0.96
Random Forest	Label	0.94	0.92	0.82	0.95	0.98
Gradient Boosting	Label	0.95	0.91	0.85	0.96	0.98
Red neuronal	Label	0.92	0.89	0.78	0.93	0.96
	One-hot	0.91	0.91	0.74	0.91	0.91

Tabla 2. Métricas de los algoritmos para predicción del alta de un paciente.

Analizando la Tabla 2, se observa como el funcionamiento general de estos modelos, excluyendo los algoritmos de Naive Bayes, es bastante bueno. Consiguiendo para esta estimación unos resultados mejores que para el anterior experimento, echo que puede ser debido a que en este caso las clases a predecir no se encontraban tan desbalanceadas, ya que se dispone de 3977 registros de la categoría 'alta' y 15675 de la categoría 'no alta'.

Aun siendo menor el efecto de la desigualdad entre clases a predecir, este efecto está presente también en este experimento, por lo que se han utilizado técnicas de oversampling [SMOTE] sobre la clase minoritaria con el fin de mejorar los resultados de los modelos, obteniendo predicciones como la mostrada en la Figura 2.

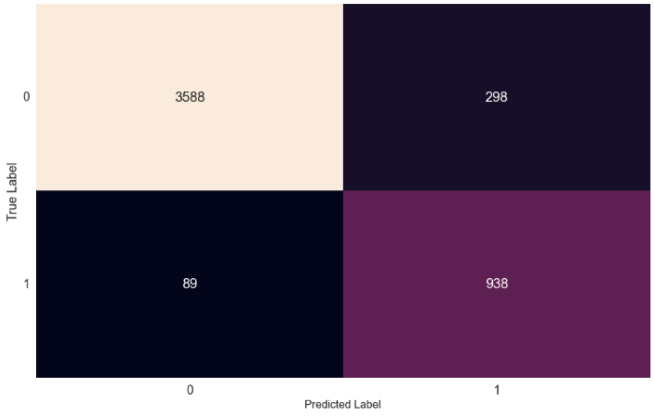


Fig. 2. Matriz de confusión para red neuronal codificación label, donde la categoría 1 representa' alta' y la 0 'no alta'.

VI. Discusión

Tal y como demuestra el trabajo es posible desarrollar software que genere datasets que pueden utilizarse para crear modelos predictivos en base a colas de mensajes HL7. Quizá estos datasets ofrezcan un desbalanceo significativo que puedan ser mejorados con técnicas de oversampling (aumento de muestras), o simplemente con históricos más grandes que permitan reducir el número de registros para balancear los datos (undersampling). Este desbalanceo es mayor cuando se utilizan datos del segmento ITR, ya que un porcentaje pequeño de pacientes recibe intervenciones quirúrgicas mientras está hospitalizado. En caso de disponer de un volumen mayor de datos sería una buena opción aislar los episodios que contengan información ITR para realizar predicciones como diagnósticos u otras variables médicas contenidas en el segemento.

Aunque 2.2M de mensajes pueden parecer muchos, es importante destacar que cada registro de episodio de paciente está compuesto por la secuencia de un conjunto de mensajes, por lo que el número de registros definitivos se reduce considerablemente.

Por otro lado, otros trabajos utilizan otro tipo de datos relacionados con constantes médicas como resultados de ciertos parámetros en analíticas y diagnósticos. Los datos propagados por cada sistema dependen de cada organización en base a las funcionalidades que implementen sus sistemas. En este caso se ha partido mayormente de datos administrativos que pudieran ser combinados con otros datos médicos con el objetivo de disponer de un conjunto mayor de variables predictoras y presumiblemente obtener mejores resultados.

VII. Conclusiones

En el presente trabajo se ha extraído información de la mensajería HL7 de un hospital, creándose dos datasets; el primero de ellos incluye toda la información relacionada con cada episodio que genera un circuito de paciente que ha sido hospitalizado o ha tenido un episodio en urgencias. Se ha creado un segundo dataset, basado en el primero, pero añadiendo registros adicionales por cada traslado que han tenido los pacientes tanto en urgencias como en hospitalización. Con esa información, se han realizado unas predicciones con el fin de determinar si los datos recogidos, utilizados originalmente para labores de gestión en el centro hospitalario, pueden ser empleados también para predecir el comportamiento futuro que pueda tener un paciente mediante técnicas de aprendizaje automático y profundo.

Se han establecido una serie de algoritmos y modelos predictivos para llevar a cabo las predicciones estimando, concretamente, el alta de un paciente (alta hospitalaria, alta voluntaria, alta por muerte, …) mediante la información de los traslados y prediciendo la posible muerte de una persona asistida en el hospital sabiendo los datos de los episodios hospitalarios y de urgencias.

Por último, se ha realizado un análisis de los resultados obtenidos por las predicciones para determinar la capacidad predictora de los datos recogidos. Del mismo modo, se han evaluado los diferentes modelos desarrollados, comparando el rendimiento dado por cada uno de ellos.

Tras efectuar todo lo anteriormente enunciado, se puede demostrar el uso de esta información para llevar a cabo estudios o implementaciones de modelos predictivos que permitan estimar diferentes variables, ya que los resultados obtenidos han sido correctos y las métricas analizadas han arrojado un buen funcionamiento de los algoritmos. Aunque, no hay que olvidar que en el caso de la predicción de alta por muerte los resultados son mejorables, por lo que sería necesario incorporar nuevas variables que tengan más capacidad predictiva y/o aumentar el dataset con más datos etiquetados con 'muerte'.

Por otro lado, y más allá de las cuestiones relacionadas con el empleo de una u otra técnica, se destaca en este trabajo que, aunque el conjunto de datos cuenta con un gran porcentaje de valores ausentes y tiene un gran desbalanceo entre las clases, los métodos que se han empleado han demostrado que se comportan muy bien y que ofrecen un gran número de posibilidades. De igual manera, se extrae de la elaboración de este trabajo la importancia de hacer un buen procesamiento de los datos. El filtrado de filas o columnas poco interesantes, la imputación de los valores ausentes o la estandarización de los datos son cuestiones que hay que tener muy en cuenta durante el proceso de la creación de los modelos.

REFERENCIAS

[1] Alekh, Sunder, S. &., & Gaur. (2015). Building a DICOM/HL7 compliant Anonymizer Service conforming to HIPAA De-identification Regulations. 10.13140/RG.2.1.2481.0967.

[2] Baxt, W. (1992). Analysis of the clinical variables driving decision in an artificial neural network trained to identify the presence of myocardial infarction. Annals of emergency medicine, 21(12). https://doi.org/10.1016/s0196-0644(05)80056-3, 1439–1444.

[3] Breiman, L. (1996). Bagging Predictors. Machine Learning, 24. 123-140.

[4] Breiman, L. (2001). Random forests. Machine learning, vol. 45(1). 5-32.

[5] Cooper, G. F., Aliferis, C. F., Ambrosino, R., Aronis, J., Buchanan, B. G., Caruana, R., . . . Spirtes, P. (1997). An evaluation of machine-learning methods for predicting pneumonia mortality. Artificial intelligence in medicine, 9(2), 107–138 https://doi.org/10.1016/s0933-3657(96)00367-3.

[6] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

[7] Hadzikadic, M. (1992). Automated design of diagnostic systems. Artificial Intelligence in Medicine. https://doi.org/10.1016/0933-3657(92)90019-L, 4(5), 329–342.

[8] Islam, M. J., Wu, Q. J., Ahmadi, M., & Sid-Ahmed, M. A. (2007). Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers. In 2007 International Conference on Convergence Information Technology (ICCIT 2007). IEEE, 1541-1546.

[9] John Lu, Z. Q. (2010). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Journal of the Royal Statistical Society Series A, 173, issue 3, 693-694.

[10] King, Z., Farrington, J., Utley, M., Kung, E., Elkhodair, S., Harris, S., . . . Crowe, S. (2022). Machine Learning for Real-Time Aggregated Prediction of Hospital Admission for Emergency Patients. EuropePMC.

[11] Kline, J. A., Courtney, D. M., Kabrhel, C., Moore, C. L., Smithline, H. A., Plewa, M. C., . . . Nordenholz, K. (2008). Prospective multicenter evaluation of the pulmonary embolism rule-out criteria. Journal of Thrombosis and Haemostasis, 6(5):772-780. https://doi.org/10.1111/j.1538-7836.2008.02944.x.

[12] Leinweber, D. (2013). Forbes. Obtenido de https://www.forbes.com/sites/davidleinweber/2013/04/26/big-data-gets-bigger-now-google-trends-can-predict-the-market/

[13] Maglogiannis, I., Karpouzis, K., Wallace, M., & Soldatos, J. (2007). Emerging Artificial Intelligence Applications in Computer Engineering - Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies. IOS Press.

[14] Marewski, J. N., & Gigerenzer, G. (2012). Heuristic decision making in medicine. Dialogues in clinical neuroscience, 14(1), 77–89. https://doi.org/10.31887/DCNS.2012.14.1/jmarewski.

[15] Miller, M. (1977). Medical Diagnostic Models: A Bibliography by M. Clinton Miller, III ... [et Al.], volume 1. University Microfilms.

[16] Morales, E., & Escalante, H. J. (2012). Aprendizaje bayesiano. Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla, México.

[17] Raita, Y., Goto, T., & Faridi, M. (2019). Emergency department triage prediction of clinical outcomes using machine learning models. Crit Care. 2019 Feb 22;23(1):64. doi: 10.1186/s13054-019-2351-7. PMID: 30795786; PMCID: PMC6387562.

[18] Reggia, J. A. (1993). Neural computation in medicine. Artificial Intelligence in Medicine, 5(2), 143–157. https://doi.org/10.1016/0933-3657(93)90014-T.

[19] Sahni, N., Simon, G., & Arora, R. (2018). Development and Validation of Machine Learning Models for Prediction of 1-Year Mortality Utilizing Electronic Medical Record Data Available at the End of Hospitalization in Multicondition Patients: a Proof-of-Concept Study. Journal of General Internal Medicine, 33(6), 921–928. doi:10.1007/s11606-018-4316-y.

[20] Silver, N. (2008). FiveThirtyEight. Obtenido de https://projects.fivethirtyeight.com/2020-election-forecast/

[21] Swain, M., & Kharrazi, H. (2015). Feasibility of 30-day hospital readmission prediction modeling based on health information exchange data. 84(12):1048-56. doi: 10.1016/j.ijmedinf.2015.09.003. Epub 2015 Sep 14. PMID: 26412010.

[22] Wu, J., Grannis, S. J., Xu, H., & & Finnell, J. T. (2016). A practical method for predicting frequent use of emergency department care using routinely available electronic registration data. 16(1), 12. doi:10.1186/s12873-016-0076-3.

[23] Zhu, K., Lou, Z., Zhou, J., Ballester, N., Kong, N., & Parikh, P. (2015). Predicting 30-day Hospital Readmission with Publicly Available Administrative Database. A Conditional Logistic Regression Modeling Approach. Methods Inf Med. 2015;54(6):560-7. doi: 10.3414/ME14-02-0017. Epub 2015 Nov 9. PMID: 26548400.