

Sistema Interactivo para el Seguimiento de Salud y Predicción de Hábitos

Pau Montesinos Caliz

Resumen—Este proyecto presenta el diseño e implementación de un modelo predictivo para la evaluación del riesgo de ictus y enfermedades cardíacas mediante el uso de técnicas de Machine Learning y Deep Learning. A través del uso de datasets de clínicos reales, se aplicaron procesos de limpieza, transformaciones y reequilibrado de balanceo de clases. Se entrenaron los modelos priorizando métricas concretas tales como recall o precision, intentando maximizar así la detección de casos positivos. Además, se integró una plataforma web mediante la cual el usuario puede introducir sus datos personales y obtener una estimación categórica del riesgo.

Palabras clave—Deep Learning, Machine Learning, predicción de enfermedades, ictus, enfermedad cardíaca, desbalanceo de clases, plataforma web, evaluación de riesgo, recall, precision

Abstract—This project presents the design and implementation of a predictive model for assessing the risk of stroke and heart disease using Machine Learning and Deep Learning techniques. Based on real clinical data, preprocessing techniques were applied, including cleaning, transformations and class imbalance adjustment. These models were trained prioritizing metrics such as recall and precision, aiming to maximize positive case detection. Additionally, a web platform was developed, allowing users to enter their personal data and receive a categorical risk estimation.

Index Terms—Deep Learning, Machine Learning, disease prediction, stroke, heart disease, class imbalance, web platform, risk assessment, recall, precision

1 INTRODUCCIÓN – CONTEXTO DEL TRABAJO

Las enfermedades cardíacas y los accidentes cerebrovasculares (ictus) representan en la actualidad una gran tasa de mortalidad a nivel mundial. La posibilidad de una detección temprana y su prevención juegan un papel clave en la mejora de la calidad de vida de los pacientes y es una forma entre muchas de reducir la masificación que sufre el sistema sanitario debido a este tipo de casos.

En este contexto, el surgimiento de la Inteligencia Artificial ha sido especialmente beneficioso, ofreciendo técnicas como Machine Learning o Deep Learning, que demuestran un gran potencial como soporte dentro del ámbito clínico.

A partir de esta base, surge la idea del proyecto, el cual tiene el objetivo de diseñar e implementar, mediante datos clínicos, algoritmos de predicción con Machine Learning y Deep Learning que evalúen el estado de salud del paciente, ofreciendo así una herramienta más que puedan usar sus usuarios y pueda ofrecer una mejora en su calidad de vida.

Además, se busca que sea una herramienta accesible para este usuario final, por lo que también surge la idea del desarrollo de una plataforma web sencilla que, con el único paso de introducir sus datos personales, el usuario pueda recibir una idea de cómo se encuentra su estado de salud en ese momento y valorar si requiere de un seguimiento médico.

2 OBJETIVOS

El objetivo final del proyecto es la creación de una plataforma web capaz de recoger los datos sobre un usuario y, a través de ellos, analizarlos y dar una estimación de la probabilidad de padecer cierta enfermedad, en este caso tomando como base un dataset de enfermedades cardíacas y enfermedades cerebrovasculares. Para llegar a cumplir este objetivo se hace un desglose en objetivos más a corto plazo como se define a continuación:

- Identificación de datasets útiles. El dataset se considerará útil en el caso de que contenga datos clínicos, hábitos y una columna con información sobre la enfermedad (la tiene/la ha tenido o no la tiene/no la ha tenido).
- Revisar los datasets escogidos y hacer limpieza de los campos y datos no aptos para el entrenamiento.
- Entrenar modelos Machine Learning y Deep Learning mediante los datos obtenidos de la limpieza realizada a los datasets. Para buscar los mejores parámetros se entrenará el mismo modelo varias veces.
- Evaluar los modelos entrenados y decidir cuál de ellos da un mejor resultado
- Diseño y creación de plataforma web de identificación de riesgos para las diferentes enfermedades descritas anteriormente.

-
- Correo de contacto: 1637556@uab.cat
 - Trabajo tutorizado por: Oriol Cortés Comellas (Área de Ciencias de la Computación e Inteligencia Artificial)
 - Curso 2024/25

3 ESTADO DEL ARTE

Dentro del contexto clínico, la inteligencia artificial ya ha comenzado a utilizarse para realizar predicciones de distintos tipos. Aún ser una herramienta relativamente reciente, su implementación ha sido muy efectiva en diversos ámbitos. De hecho, ya se han aplicado modelos predictivos en algunas enfermedades, demostrando así su efectividad dando soporte en la toma de decisiones médicas.

3.1 Predicción de Ictus

Relacionado a la predicción de Ictus se han lanzado proyectos como Innostroke. Este proyecto basado en inteligencia artificial tiene como objetivo monitorizar y prever el ictus, una de las principales causas de muerte a nivel mundial. Para ello, la tecnología desarrollada se basa tanto en el uso de electrocardiogramas como en datos de estilo de vida recogidos a través de dispositivos inteligentes. [1]

3.2 Predicción de enfermedades al corazón

Mediante el uso de una radiografía de tórax, un grupo de investigadores liderados por Jakob Weiss, radiólogo del Centro de Investigación de Imágenes Cardiovasculares del Hospital General de Massachusetts, ha entrenado un modelo de predicción dedicado a predecir el riesgo de muerte tanto por ataque cardíaco como por ictus en los siguientes 10 años. Los resultados de este modelo fueron presentados en la reunión anual de la Sociedad Radiológica de América del Norte. [2]

4 METODOLOGÍA

Para la realización del proyecto se ha optado por el uso de una metodología ágil, concretamente Kanban, con la cual el flujo de trabajo se ha dividido en diferentes partes para una mayor eficiencia global. A través de un sistema visual de tableros, en este caso Trello [3], se han descompuesto en pequeños bloques los pasos para llegar al objetivo final, asignando así a cada uno de ellos una tarea para cumplir los objetivos parciales definidos en el apartado 2. Como complemento a la planificación, se ha incluido un diagrama en el anexo A1 definiendo la distribución del proyecto.

Desde un punto de vista más técnico, el proyecto se ha estructurado de una forma modular, permitiendo así a cada componente ser tanto independiente como reutilizable. Mantener la estructura ofrece una mejora en el código en legibilidad y mantenimiento. Además, también ayuda en la escalabilidad y la implementación de cambios futuros sin comprometer la integridad del resto del sistema.

A lo largo del desarrollo se han utilizado múltiples herramientas, seleccionadas para cada una de las tareas a realizar. A continuación, se detallan las más relevantes de todas ellas:

- Desarrollo y procesamiento de datos:
 - Python: Sistema principal de desarrollo, ideal para el proyecto debido a la gran cantidad de librerías dedicadas a la ciencia e ingeniería de datos
 - Scikit-learn [4]: Biblioteca usada para el entrenamiento y la validación de los modelos de Machine Learning.
 - XGBoost [5]: Biblioteca elegida para el entrenamiento de modelos de Machine Learning basados en boosting.
 - Keras [6]: Biblioteca de alto nivel basada en TensorFlow usada para el desarrollo de los modelos de Deep Learning.
 - Imblearn [7]: Biblioteca utilizada para la implementación de métodos de balanceo de clases.
- Infraestructura y despliegue:
 - PyCharm: Entorno de desarrollo del código. Escogido por la facilidad de integración con Git y navegación por los diferentes módulos.
 - Github: Repositorio de código usado para mantener el código a salvo y en la última versión
 - Google Colab: Plataforma Cloud con acceso a GPU. Usada para el entrenamiento de modelos en la fase de Deep Learning.
- Plataforma Web:
 - Flask [8]: Microframework web utilizado para el despliegue de la plataforma web.
 - HTML y CSS [9] [10]: Lenguajes utilizados para la creación de la interfaz de la plataforma web.

5 DESARROLLO

Para comenzar el proyecto, se ha creado un repositorio en GitHub, conteniendo todo el código actualizado. El desarrollo del código se hace en la rama de develop y, en el momento de tener un código madurado, se hace un merge a la rama definitiva main.

Dentro del repositorio se pueden apreciar diferentes carpetas y paquetes de Python:

- Datasets: incluye los datasets que se utilizarán en el proyecto en formato csv.
- Deprecated: incluye código creado que finalmente no tiene uso.
- Development: incluye el código que realiza el entrenamiento de los modelos de Machine Learning y diferentes paquetes de python de los cuales este código hace uso:

- Clean_datasets: Incluye los archivos de código de limpieza de los datasets.
- ML_models: Incluye los archivos de creación de las clases utilizadas como base de los modelos de Machine learning.
- Pruebas: algunas pruebas realizadas al código creado.
- Web_platform: Incluye dos carpetas y dos archivos necesarios para lanzar la aplicación:
 - Templates: incluye las plantillas html utilizadas para la visualización de la plataforma web.
 - Static: incluye los archivos css que se encargan de darle estilo a las plantillas HTML.
 - Riesgos.py: Archivo formado por funciones que calculan los riesgos de padecer cierta enfermedad a través de los datos introducidos por el usuario.
 - App.py: Archivo que contiene el código desarrollado con Flask dedicado al despliegue de la plataforma web en servidor local.

5.1 Datasets utilizados

Para llevar a cabo el desarrollo del proyecto se inició con un total de 3 datasets, siendo uno de ellos relegado a mitad de camino debido a la naturaleza del mismo.

Estos 3 datasets son:

- EnfermedadCorazon.csv: Este dataset recoge el registro de datos y hábitos de diferentes usuarios (18 campos en total por cada registro). Estos datos incluyen tanto hábitos, como si es fumador o bebedor, como datos personales o clínicos, como el sexo de la persona, la edad o el padecimiento de alguna otra enfermedad como asma. El dataset también incluye un campo indicando si la persona padece o no una enfermedad del corazón, importante para evaluar más tarde el rendimiento de los modelos.
- Stroke.csv: Este dataset, al igual que al anterior, recoge datos y hábitos de diferentes usuarios, contando en este caso con un total de 12 campos por registro. En este caso también se contemplan hábitos y datos personales y clínicos, como podrían ser el estado de fumador, la edad o la presencia de hipertensión.
- Cancer.csv: Este dataset es muy diferente a los demás. Mientras que en los otros dos se pueden ver registros que acaban en la presencia o no de cierta

enfermedad, en este se encuentran un total de 31 columnas indicando las características de un cáncer concreto. Debido a esto, el objetivo que se tendría con este dataset de información no es el de predecir si padece cierta enfermedad o no, sino el de predecir si el cáncer sufrido por el usuario relacionado con el registro es benigno o maligno. Entre los campos de este dataset se encuentran parámetros como radio, textura o área. Debido a esta información, este es el dataset que se ha decidido descartar.

5.2 Limpieza de datasets

Para cada uno de los datasets empleados, se ha diseñado un proceso de limpieza específico adaptado a las características particulares de los datos con el objetivo de facilitar el entrenamiento de los modelos. La limpieza aplicada a cada dataset fue:

- Cancer.csv:
 - Se cambian los valores de *diagnosis* M y B (Maligno y Benigno) por 1 y 0.
 - Los valores 0.0 se convierten en NA y posteriormente se eliminan
- Stroke.csv:
 - Se cambian los valores de *smoking_status* por -1, 0, 1 y 2, siendo el -1 el estado de Unknown.
 - Se rellenan los NA dentro de la columna *bmi* por su mediana.
 - Se elimina la columna *id* ya que no tiene ningún tipo de aporte.
 - Se factorizan las columnas *gender*, *Residence_type* y *work_type*.
- HeartDisease.csv
 - Se cambian los valores de rango de edad por el valor mediano del rango.
 - Se mapean las columnas *GenHealth* y *Diabetic* cambiando sus valores en orden ascendente según el valor de cada fila.
 - Las columnas *Smoking*, *AlcoholDrinking*, *HeartDisease*, *Stroke*, *SkinCancer*, *KidneyDisease*, *Asthma* y *DiffWalking* se mapean según si son yes (1) o no (0).
 - Las columnas *Sexo* y *Raza* se factorizan.

5.3 Modelos de Machine Learning creados

Se han creado un total de 3 modelos de Machine Learning, cada uno de ellos representado mediante una clase de Python. Los métodos que tienen las clases son compartidos, es decir, que lo único que cambia entre ellas son los parámetros con los cuales se entrena. Los métodos compartidos son:

- Train: Entrena el modelo con diferentes parámetros buscando los mejores y posteriormente devuelve el modelo entrenado que haya obtenido los mejores resultados. Esta búsqueda de parámetros se hace mediante una *Grid Search* (entrena modelos con diferentes parámetros para encontrar el que mejores resultados da)
- Predict: Predice los valore de un conjunto de datos dado. Este método proporciona un fallo si el modelo no ha sido entrenado previamente.
- Predict_proba: Devuelve la probabilidad de que una muestra de un conjunto de datos pertenezca a cada una de las clases posibles dentro de un modelo previamente entrenado.
- Evaluate: Dado un conjunto de datos hace su predicción y evalúa el resultado. Dada una predicción, evalúa directamente el resultado de esa predicción.
- Save_model: Guarda el modelo entrenado en un archivo pickle.
- Load_model: Carga un modelo entrenado y guardado previamente.

A continuación, se especifican los modelos que han sido creados y la razón por la cual han sido escogidos:

- Logistic Regression:
 - Modelo rápido y fácil de entrenar.
 - Modelo simple que se puede usar como *baseline* para comparar con los demás.
- Random Forest:
 - Al entrenar diferentes árboles y en cada uno de ellos conjuntos aleatorios de datos, el overfitting se reduce de manera considerable.
 - Al combinar distintos árboles entre sí, pequeños errores en los datos que puedan pasar en cada uno de ellos se compensan.
- XGBoost
 - Este es uno de los modelos más potentes actualmente y con mayor rendimiento predictivo.
 - Cada nuevo árbol es entrenado para corregir el error de los anteriores.
 - Suele ofrecer precisiones superiores a los demás modelos de Machine Learning.

5.4 Modelos de Deep Learning creados

Se ha creado un modelo de Deep Learning el cual poder entrenar con los datasets que tenemos. Este modelo creado es una MLP (Multilayer Perceptron) formado por una capa de entrada, varias capas ocultas y una capa de salida. En el

caso específico de este modelo, se empieza por una capa *dense* (capa densa totalmente conectada) y se continua por una sucesión de capas con función de activación *relu* (convierten los negativos a 0), aunque también se han probado las capas *leakyRelu* (en vez de convertir los negativos a 0 los dejan en un número cercano a este) con resultados similares. Por último, se acaba con una función de activación *sigmoid*, la cual recoge el resultado de las anteriores capas y las convierte en una probabilidad que va de 0 a 1, permitiendo así más tarde añadir un *threshold* que divida los resultados.

Entre las diferentes capas del modelo se encuentran diferentes técnicas orientadas sobre todo a la generalización de los resultados, reducción del overfitting y optimización del entrenamiento. Entre ellas destacan:

- Dropout: Durante el entrenamiento, cuando se aplica esta técnica de regularización, se desactivan aleatoriamente cierto número de neuronas (solo durante esa iteración del entrenamiento). Esto evita que el modelo cree una dependencia sobre neuronas específicas, ayudando así al modelo a una mayor generalización.
- Batch Normalization: Batch normalization es una técnica que ayuda al modelo a estabilizar y acelerar el entrenamiento. Para hacer esto normaliza las activaciones de una capa, es decir, transforma los valores de la capa para que tengan una distribución controlada y estable. Esta técnica se aplica normalmente antes de la función de activación, haciendo que esta reciba una entrada con una media cercana al 0 y una varianza controlada.
- Regularización: Penaliza los pesos grandes. Esta técnica ayuda al modelo a generalizar más. Esto lo consigue no memorizando sino aprendiendo lo esencial, pudiendo hacer más tarde una generalización aplicando rasgos generales y no específicos de algunos de los registros.

La estructura con la cual se ha trabajado se puede ver de manera más precisa en esta imagen:

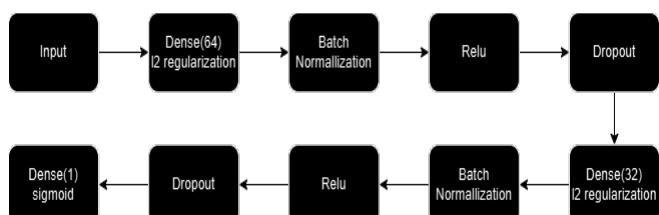


Figura 1. Estructura del modelo de Deep Learning

5.5 Técnicas de control de desbalanceo de clases

En un previo análisis de los datasets escogidos finalmente, se ha podido denotar una característica notoria en ambos: un claro desbalanceo de clases. Este gráfico de barras muestra cuan desbalanceada está la información que se trata:

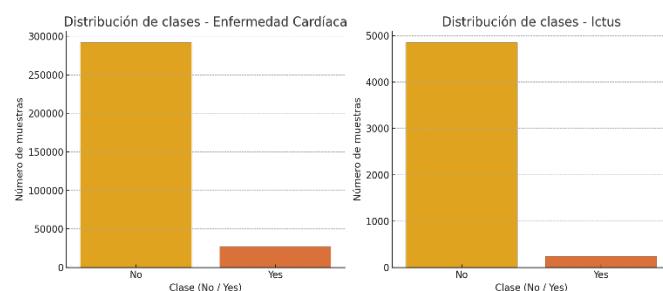


Figura 2. Distribución de clases de los datasets

En realidad, aunque la diferencia parezca muy grande, se asemeja en gran medida con la realidad, en la cual la mayoría de personas están sanas y solo una minoría sufre de estas enfermedades. La relación precisa de desbalanceo de cada uno de los datasets es de 1:10.87 para el de enfermedades cardíacas y de 1:19.52 para el de enfermedades cerebrovasculares (ictus).

Para resolver el problema del desbalanceo de clases se han aplicado varias técnicas por separado, entrenando los modelos con cada una de ellas para ver con cual se llegaban a obtener mejores resultados. A continuación, se describen las técnicas mencionadas y como se han definido:

- Random oversampling: Duplicado aleatorio de las filas de la clase minoritaria hasta llegar al mismo número de registros que la clase mayoritaria.

```
def over_sampling_technique(X_train, y_train):
    from imblearn.over_sampling import RandomOverSampler
    ros = RandomOverSampler(random_state=50)
    X_train, y_train = ros.fit_resample(X_train, y_train)
    return X_train, y_train
```

- SMOTE: Parecido al oversampling pero, en vez de la duplicación de filas, hace uso de la interpolación con los vecinos más cercanos para la creación de nuevas filas. También se hizo uso de la variante SMOTENC la cual trata de manera diferente a datos categóricos y numéricos, más dieron resultados muy similares.

```
def SMOTE_technique(X_train, y_train):
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=50)
X_train, y_train = smote.fit_resample(X_train, y_train)
return X_train, y_train
```

- Random Undersampling: Reducción aleatoria de registros de la clase mayoritaria. Esto hace que el modelo esté balanceado, sin datos duplicados ni sintéticos. A cambio, se obtiene una reducción muy significativa de la cantidad de datos.

```
def under_sampling_technique(X_train, y_train):
    from imblearn.under_sampling import RandomUnderSampler
    rus = RandomUnderSampler(sampling_strategy='auto', random_state=50)
    X_resampled, y_resampled = rus.fit_resample(X_train, y_train)
    return X_resampled, y_resampled
```

- Tomek Links: Undersampling eliminando registros de la clase mayoritaria mediante la distancia euclíadiana. No balancea las clases, solo elimina los puntos más cercanos entre clases para que haya una mejor diferenciación entre ellas.

```
def tomek_links_technique(X_train, y_train):
    from imblearn.under_sampling import TomekLinks
    tomek = TomekLinks(sampling_strategy='auto')
    X_resampled, y_resampled = tomek.fit_resample(X_train, y_train)
    return X_resampled, y_resampled
```

- SMOTETomek: Undersampling eliminando registros cercanos mediante Tomek Links y posteriormente aplicación de SMOTE para ampliar los registros de la clase minoritaria.

```
def smotetomek_technique(X_train, y_train):
    from imblearn.combine import SMOTETomek
    smote_tomek = SMOTETomek(random_state=50)
    X_train, y_train = smote_tomek.fit_resample(X_train, y_train)
    return X_train, y_train
```

- Establecimiento de pesos: Aumento del peso de los errores de la clase minoritaria. Siendo 0 la

clase mayoritaria y 1 la clase minoritaria, un peso de 1:2 haría que se penalizase el doble los errores cometidos en la clase 1.

```
model.fit(
    X_train, y_train,
    validation_split=0.1,
    epochs=100,
    batch_size=32,
    class_weight={0: 1, 1: 1},
    callbacks=[early_stop],
    verbose=1
)
```

En este fragmento de código también se pueden apreciar más parámetros interesantes:

- validation_split: Especifica qué porcentaje de los datos va a ir al conjunto de validación y, por tanto, indirectamente, qué porcentaje va a ir al conjunto de train
- epochs: Indica cuantas veces se va a recorrer el conjunto de entrenamiento durante el aprendizaje del modelo.
- batch_size: Define la cantidad de datos que utiliza el modelo antes de actualizar los pesos.
- class_weight: Parámetro que asigna a cada clase un peso, compensando así el desbalanceo de clases haciendo que el modelo tenga más en cuenta la clase minoritaria.
- callbacks: Parámetro que sirve como mecanismo para ejecutar diferentes funciones automáticamente en ciertos momentos del entrenamiento como podrían ser al final de una epoch o al acabar el entrenamiento. En este ejemplo, early stop acaba el entrenamiento si ve que cierta métrica no mejora.

5.6 Visualización de los datos

Para realizar la visualización de los datos y decidir cuál es el mejor resultado, se han decidido crear dos visualizaciones.

En primer lugar, una matriz de confusión viendo donde se ha clasificado cada uno de los resultados.

Teniendo esta imagen, se podría saber en cada momento la distribución de resultados y como mejorar el modelo.

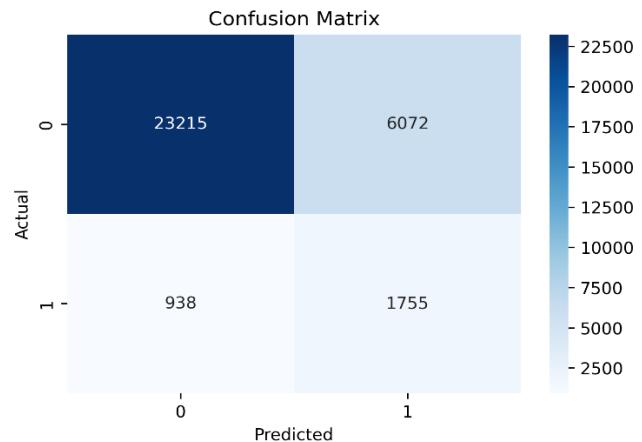


Figura 3. Ejemplo de matriz de confusión de predicciones

En segundo lugar, otra matriz de confusión mostrando las métricas principales del entrenamiento del modelo calculadas sobre la matriz de confusión inicial.

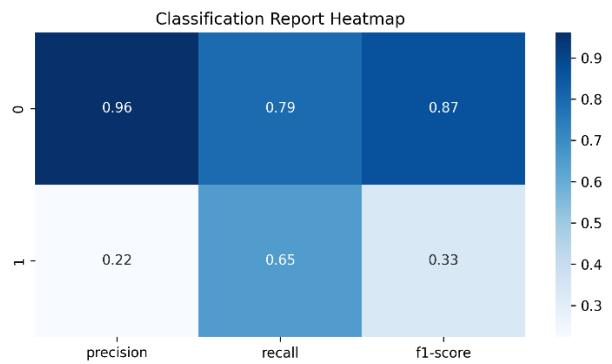


Figura 4. Ejemplo de matriz de confusión de métricas

5.7 Plataforma web

Para finalizar el proyecto, se ha creado una plataforma web capaz de calcular, a través de los datos introducidos por los usuarios el riesgo de padecer las enfermedades que estamos tratando. Cabe destacar que la plataforma web no es alimentada por los modelos predictivos calculados anteriormente, sino que se hace un cálculo definido manualmente. Esto es a causa de los resultados que se mencionarán posteriormente.

Al entrar dentro de la plataforma, se puede ver una pantalla en la cual se le puede dar a un botón para ir al menú principal.



Figura 5. Menú de inicio de la plataforma web

Posteriormente, se entra dentro del menú principal, en el cual el usuario podrá decidir si evaluar los riesgos de tener un ictus o el de tener una enfermedad cardíaca.



Figura 6. Menú principal de la plataforma web

Dentro de los dos apartados de evaluación de riesgos, se verá un formulario en cada uno, en el cual el usuario deberá añadir sus datos.

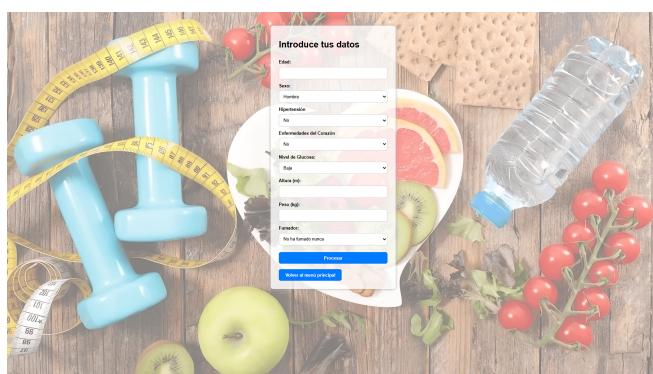


Figura 7. Formulario de datos de la plataforma web

Por último, al introducir los datos, la plataforma enviará un mensaje dependiendo de los cálculos que haya hecho con los datos introducidos.

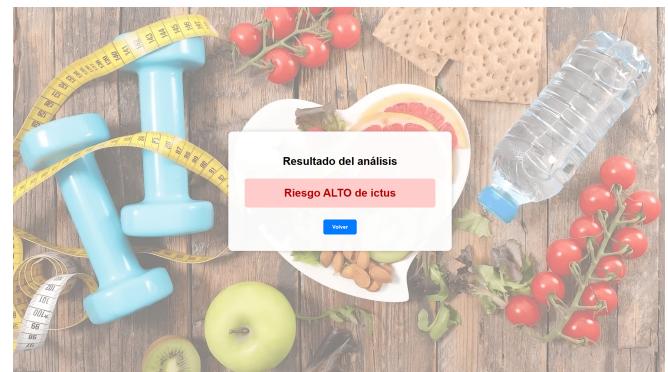


Figura 8. Mensaje de riesgo de la plataforma web

La plataforma ha sido implementada usando el micro-framework de Python Flask. Acompañando a este framework está definida una sucesión de archivos HTML, los cuales definen la estructura de las vistas, y archivos CSS, los cuales definen el diseño visual. La lógica que define el cálculo de los riesgos está integrada en el backend. Esta lógica está hecha a partir de un análisis de los parámetros que más tienen los pacientes que sí sufren de la enfermedad. La lógica recibe los datos introducidos en los formularios, los procesa y genera la respuesta correspondiente.

6 RESULTADOS

En general, los resultados de las predicciones mediante el uso de los modelos y técnicas mencionadas anteriormente no han sido positivos. Aún haber sido aplicados uno por uno con los mejores parámetros, ni los modelos de Machine Learning ni los modelos de Deep Learning han sido capaces de detectar un patrón por el cual ofrecer una predicción acertada. Los resultados se han dividido entre clase minoritaria y clase mayoritaria para evitar problemas debido al desbalanceo de clases.

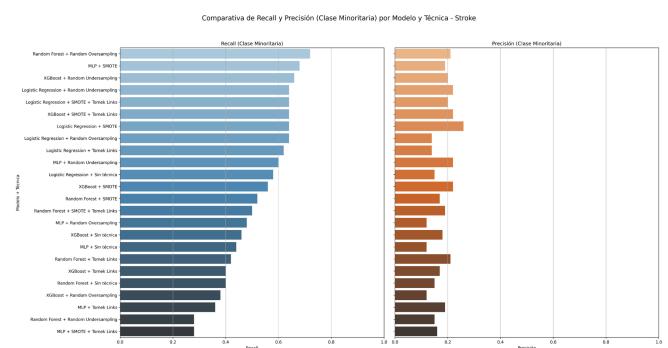


Figura 9. Comparativa de Recall y Precision de la clase minoritaria en el dataset Stroke

En este gráfico, el cual muestra la comparativa de recall y precision de la clase minoritaria dentro del dataset de stroke, aplicando diferentes modelos y técnicas de balanceo, se puede observar que, si bien la recall alcanza resultados relativamente altos (0.72 en el mejor de ellos), la

precision se mantiene baja en todas las combinaciones, teniendo en el mejor de los casos un valor de 0.26.

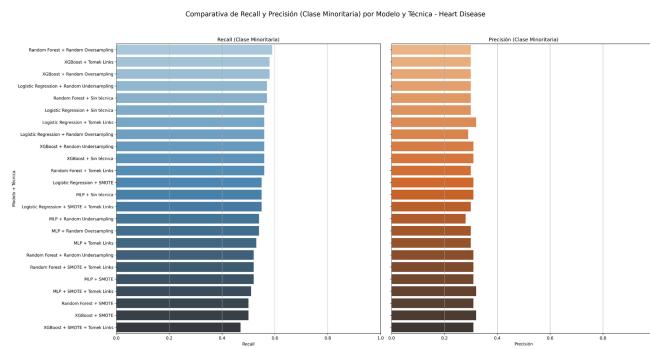


Figura 10. Comparativa de Recall y Precision de la clase minoritaria en el dataset HeartDisease

Vemos una situación similar si cambiamos al dataset de enfermedades al corazón. Las recall de la clase minoritaria son algo elevadas, más cuando pasamos a sus precisiones vemos un patrón similar al anterior dataset, aunque las precisiones también algo más elevadas debido a la mayor cantidad de información que contiene el dataset con respecto al anterior.

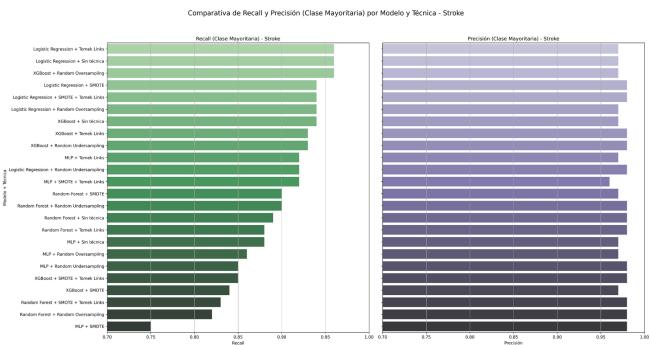
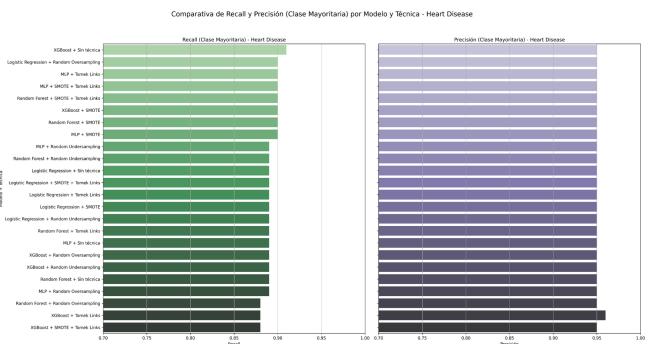


Figura 11. Comparativa de Recall y Precision de la clase minoritaria en el dataset Stroke



Figuras 12. Comparativa de Recall y Precision de la clase minoritaria en el dataset HeartDisease

Como se observa en las dos gráficas anteriores, en la clase mayoritaria, tanto la recall como la precision tienen tendencia a obtener resultados elevados en la mayoría de combinaciones. Esto nos viene a decir que, si no desglosáramos los resultados por clase, obtendríamos unas métricas globales engañosamente buenas, ocultando de esta

manera el bajo rendimiento del modelo en la detección de la clase minoritaria, que en el contexto clínico en el que hablamos, resulta ser la más relevante.

Estos resultados que se han obtenido se pueden ver de igual manera mediante una métrica que engloba las dos usadas anteriormente: f1-score. Esta métrica es calculada mediante la siguiente fórmula:

$$\text{F1-Score} = (2 \cdot \text{Precision} \cdot \text{Recall}) / (\text{Precision} + \text{Recall})$$

Como se puede observar, si una de las dos métricas es baja, el F1 será igualmente bajo y se verá que tan buena es la predicción en realidad

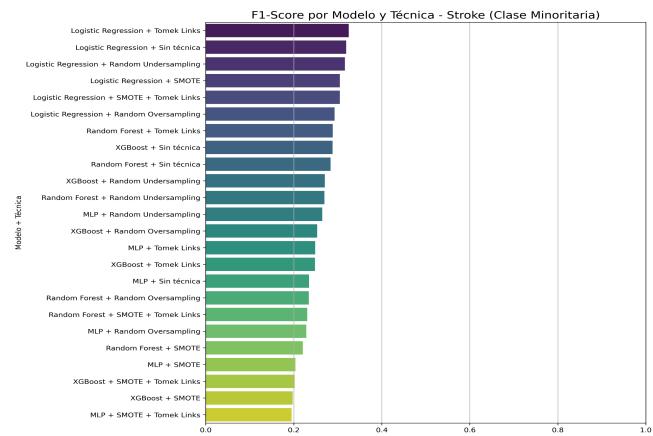


Figura 13. F1-score de la clase minoritaria en el dataset Stroke

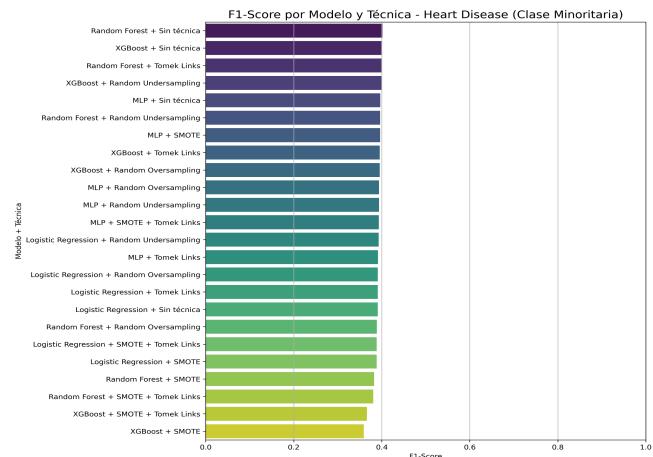


Figura 14. F1-score de la clase minoritaria en el dataset HeartDisease

Como se puede comprobar, la métrica f1-Score para la clase minoritaria es muy reducida en los dos datasets, alcanzando el 0.4 en el dataset de enfermedades al corazón y alcanzando apenas el 0.3 en el de Ictus. Este resultado refuerza la idea de que los modelos no dan una predicción óptima.

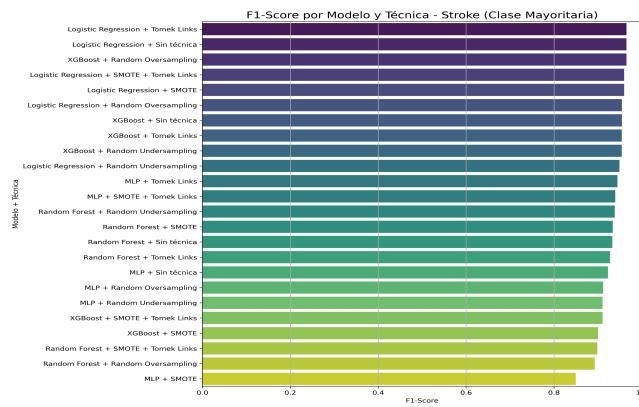
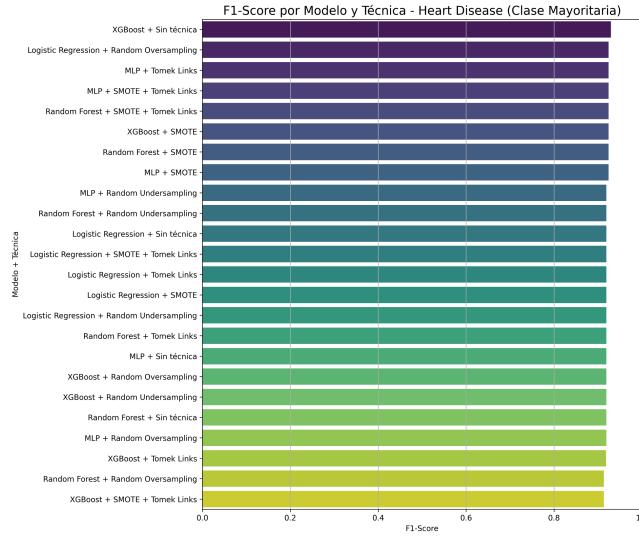


Figura 15. F1-score de la clase mayoritaria en el dataset Stroke



Figuras 16. F1-score de la clase mayoritaria en el dataset HeartDisease

Como se puede apreciar también en estas gráficas, igual que se ha apreciado anteriormente con la comparación de recall y precision, las predicciones en la clase mayoritaria están, en la mayoría de casos, por encima del 0.9, elevando las métricas globales de precisión y por tanto haciéndolas inservibles a la hora de evaluar los modelos.

6.1 Resultados erróneos identificados

Durante el desarrollo de código, surgió un error crucial, solventado posteriormente, que llevó a unos resultados que mostraban una gran mejoría.

Este problema surgido en el proceso es conocido como Data Leakage y fue dado en el momento de aplicar la técnica SMOTE. Esta fue aplicada antes de realizar la división entre train y test, provocando una contaminación del conjunto de train con información sintética generada a

partir de datos que más tarde también estarían en el conjunto de test.

Esta contaminación provocada por el Data Leakage (conocido también como fuga de datos) infló los resultados al introducir en el entrenamiento información derivada del conjunto de test, información la cual el entrenamiento debía desconocer en todo momento.

| | Precision | Recall |
|-----------|-----------|--------|
| No stroke | 0.84 | 0.72 |
| Stroke | 0.76 | 0.87 |

Tabla 1. Métricas asociadas al Data Leakage

Como se observa en esta tabla, aún ver una reducción en la clase mayoritaria, se ve una estabilización en cuanto a métricas globales. Dado que en todo momento se ha estado hablando sobre un contexto clínico, la correcta detección de la clase 1 (positiva) resulta prioritaria.

Como se ha mencionado anteriormente, por tanto, debido a la fuga de datos, estos resultados no serían aplicables a un entorno real, pero enseñan como pequeños cambios o errores pueden llevar a grandes diferencias.

7 CONCLUSIONES

A pesar de haber aplicado un gran seguido de modelos y técnicas, no se han podido observar mejoras en las predicciones que se han realizado. El aumento de métricas como la *recall* vemos como agravan la situación de otras, como la *precision*, lo que puede sugerir que el claro desbalanceo de clases no es la única limitación al hacer las predicciones.

Al fin y al cabo, este proyecto también muestra uno de los factores fundamentales a resolver dentro del desbalanceo de clases: como hacer para aumentar ciertas métricas sin hacer decaer las demás. En un contexto clínico, un par de falsos positivos puede ser insignificante si a costa de ello se puede aumentar en gran medida el número de casos correctamente predichos, pero este no es el caso al que finalmente se ha llegado, en el cual el número de malas predicciones es tan grande que tendría afectación en un rendimiento global.

El análisis de resultados realizado también deja algunos puntos relevantes. Sin embargo, uno de ellos destaca entre los demás. El no tratar los datos de forma global ha sido uno de los puntos que ha hecho que se viera que los resultados no eran buenos y, por tanto, no aplicables en un caso real. En el caso de escoger la precisión global, se vería en muchos casos un dato de precisión mayor a 0.9. Esto no quiere decir de ninguna manera que el modelo dé resultados adecuados, sino que, al ser la clase mayoritaria de un tamaño muy superior a la minoritaria, esta coge un peso relevante en la medición.

En conclusión, los resultados obtenidos reflejan las dificultades que se padecen en un problema donde hay un fuerte desbalanceo de datos. Aún haber probado gran cantidad de modelos y técnicas, no se ha podido lograr un rendimiento satisfactorio, pudiendo comenzar a surgir la idea de que habría más problemas a los que enfrentarse, como la falta de datos o campos relevantes de información.

AGRADECIMIENTOS

Quiero ofrecer mi más sincero agradecimiento a la Universidad Autónoma de Barcelona, particularmente a la Escuela de Ingeniería, por ofrecerme toda la formación y recursos académicos que han podido hacer posible la realización de este proyecto.

Agradezco también en gran medida la orientación y apoyo que me ha ofrecido mi tutor de TFG. No solo me ha ayudado a resolver dudas técnicas, sino que también me ha ayudado a mejorar la estructura general de mi proyecto.

Por último, me gustaría agradecer a mis compañeros de clase, los cuales han contribuido a mi crecimiento tanto académico como personal.

BIBLIOGRAFIA

- [1] Barcelona Supercomputing Center, "El BSC desarrolla un modelo de IA para predecir el riesgo de ictus mediante dispositivos móviles," BSC-CNS, 2023. <https://www.bsc.es/es/noticias/noticias-del-bsc/el-bsc-desarrolla-un-modelo-de-ia-para-predecir-el-riesgo-de-ictus-mediante-dispositivos-m%C3%B3viles> (accessed Jun. 15, 2025).
- [2] S. Parra, "Predicen el riesgo de derrame cerebral y ataque al corazón a partir de una radiografía," National Geographic España, Dec. 05, 2022. https://www.nationalgeographic.com.es/ciencia/la-inteligencia-artificial-va-puede-predecir-el-riesgo-de-sufrir-un-ataque-al-corazon_19183 (accessed Jun. 15, 2025).
- [3] Atlassian, "Captura, organiza y aborda tus tareas pendientes en cualquier lugar | Trello," Trello.com, 2025. <https://trello.com/> (accessed Mar. 01, 2025).
- [4] S. Developers, "scikit-learn: machine learning in Python – scikit-learn 1.6.1 documentation," Scikit-learn.org, 2025. <https://scikit-learn.org/stable/> (accessed Mar. 05, 2025).
- [5] Xgb. Developers, "XGBoost Documentation – xgboost 3.0.0 documentation," Readthedocs.io, 2025. https://xgboost.readthedocs.io/en/release_3.0.0/ (accessed Mar. 24, 2025).
- [6] K. Team, "Keras documentation: Keras 3 API documentation," Keras.io, 2025. <https://keras.io/api/> (accessed Mar. 05, 2025).
- [7] The imbalanced-learn developers., "imbalanced-learn documentation – Version 0.13.0," Imbalanced-learn.org, 2024. <https://imbalanced-learn.org/stable/> (accessed Apr. 23, 2025).
- [8] P. Projects, "Welcome to Flask – Flask Documentation (3.1.x)," Palletsprojects.com, 2025. <https://flask.palletsprojects.com/en/stable/> (accessed Mar. 05, 2025).
- [9] WHATWG, "HTML Standard," Whatwg.org, 2025. <https://html.spec.whatwg.org/> (accessed Mar. 05, 2025).
- [10] W3C, "Cascading Style Sheets," W3.org, 2024. <https://www.w3.org/Style/CSS/> (accessed Mar. 05, 2025).

ANEXO

A1. DIAGRAMA DE GANTT

