Enginyeria
UAB Universitat Autònoma de Barcelona

Universidad Autónoma de Barcelona

Escuela de Ingeniería

Grado en Ingeniería de Datos

Sistema Interactivo para el Seguimiento de Salud y Predicción de Hábitos

Alumno: Pau Montesinos Cáliz

Tutor: Oriol Cortés Comellas

Fecha: 25/05/2025

1. Desarrollo

El desarrollo de esta parte del proyecto ha sido en base a la creación y entrenamiento de un modelo de aprendizaje profundo (Deep Learning). En un principio, se había planteado la creación de dos modelos distintos (una MLP y una Sparse Attention-Based Neural Network). Debido a los malos resultados obtenidos con la MLP, derivados en gran medida de un fuerte desbalanceo de las clases en los datos, se ha optado por continuar el desarrollo exclusivamente con esta arquitectura, incorporando diversas estrategias para mejorar su rendimiento.

Para esto, se han intentado aplicar varias técnicas por separado:

- Random Oversampling: Copia aleatoria de 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: Técnica parecida al oversampling pero, en vez de duplicar filas, utiliza a los vecinos más cercanos para la generación de nuevos datos mediante la interpolación.

```
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: Técnica que reduce aleatoriamente los registros de la clase mayoritaria hasta igualar a los de la clase minoritaria, haciendo que el modelo esté balanceado con menos datos, pero con datos reales no sintéticos y sin duplicados.

```
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: técnica basada en undersampling, eliminando registros de la clase mayoritaria mediante el cálculo de la distancia euclidiana. No balancea las clases, solo elimina los puntos más cercanos en la clase mayoritaria.

```
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
```

- 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 penalizase el doble los errores 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
  - validation_split: Mide cuanta cantidad de datos va a ir a train y cuanta a validation.
  - epochs: Cantidad de veces que se va a entrenar el modelo con los datos.
  - batch_size: Cantidad de datos se van a coger a la vez.
  - class_weight: Parámetro que nos dice cual va a ser el peso de cada clase.
  - callbacks: Parámetro que hace las llamadas que contiene en su interior, en este caso un early stop, el cual para el entrenamiento, si ve que cierta métrica no aumenta, lo termina.

  En el caso del proyecto actual, los pesos se calculan automáticamente dependiendo del balanceo de las clases.

- Establecimiento de un threshold: A través de la imposición de un threshold a las probabilidades calculadas por el MLP, se puede reducir el punto de corte entre clase 0 y clase 1. Se calcula el valor f1 para todos los threshold y se elige el que mejor resultado da.

```
y_proba = model.predict(X_test)
precision, recall, thresholds = precision_recall_curve(y_test, y_proba)
f1_scores = 2 * (precision * recall) / (precision + recall + 1e-10)
best_threshold = thresholds[np.argmax(f1_scores)]
y_pred = (y_proba >= best_threshold).astype(int)
```

Además de esta decisión de solo adaptar un modelo, también se ha eliminado uno de los datasets (Cancer). Este dataset ya se había puesto en cuestión anteriormente, más ha sido en este tramo del proyecto cuando se ha decidido eliminar completamente.

Esto ha resultado así debido a la naturaleza del dataset. Mientras que los datasets relacionados a los derrames cerebrales y las enfermedades del corazón representan tanto hábitos como datos de usuarios, el de cáncer nos muestra datos sobre un cáncer en concreto. Por otro lado, vemos que las columnas objetivo también son distintas. Mientras en los dos datasets escogidos el objetivo es el saber si ha tenido o va a tener cierta enfermedad, en el de cáncer el objetivo es ver si ese cáncer es benigno o maligno.

Dados estos hechos, y debido también a la naturaleza del proyecto (salud y hábitos), se ha considerado que eliminar este dataset de las variables a tener en cuenta era la mejor opción.

Dentro del modelo escogido también se pueden ver diferentes estructuras desarrolladas en su creación. Aún esto, se puede ver una base simple: Una sucesión de capas con función de activación relu (convierte negativos a 0) acabando en una capa con función de activación sigmoid, convirtiendo el resultado en una probabilidad (rango de 0 a 1).

Entre las capas de las diferentes estructuras hat diferentes técnicas orientadas, sobretodo, a la generalización, reducción de overfitting y optimización del entrenamiento:

- Dropout: Apaga neuronas aleatoriamente. Evita dependencias a neuronas específicas y ayuda al modelo a generalizar más.
- Batch Normalization: Normaliza la salida de las capas antes de pasar a la siguiente capa.
- Regularización: Penaliza los pesos grandes. Ayuda a que el modelo generalice más, no memorizando sino aprendiendo lo esencial para poder hacer más tarde la clasificación.

Esta es la estructura sobre la cual se ha trabajado:

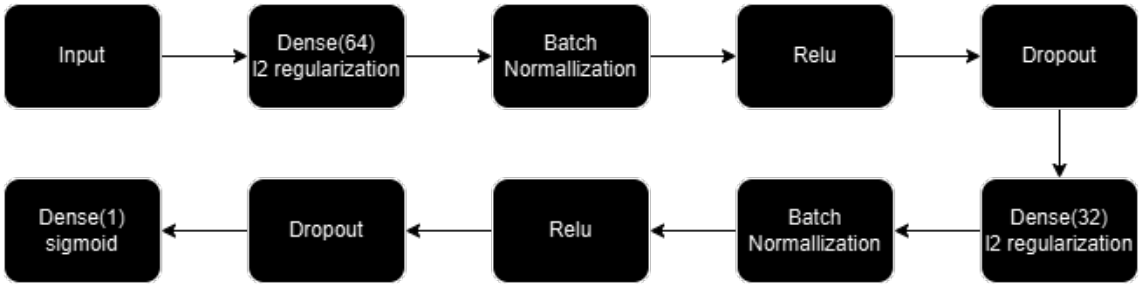

Figura 1. Estructura del modelo de Deep Learning

Por último, se ha empezado a desarrollar una plataforma la cual permita a los usuarios introducir sus datos y estimar la probabilidad de tener cierta enfermedad. Esta plataforma aún está en la fase inicial. Estas son sus primeras partes:

Figura 2. Menú principal de la plataforma web

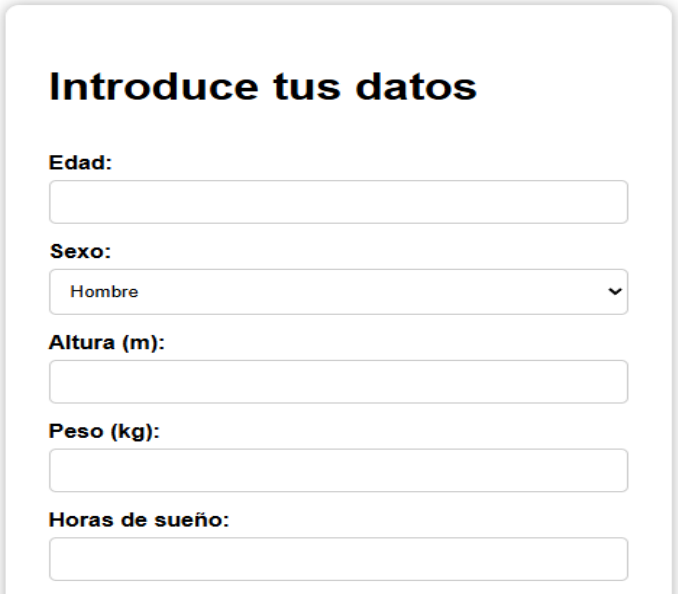

Figura 3. Formulario de datos de la plataforma web

2. Resultados previos

Los resultados obtenidos a través de los métodos descritos anteriormente no han sido positivos. Aún la aplicación de las técnicas mencionadas anteriormente, ni los modelos de Machine Learning ni los modelos de Deep Learning entrenados hasta el momento han sido capaces de detectar un patrón por el cual hacer una buena predicción. Las métricas que se han usado para medir si el modelo tiene un buen rendimiento o no son:

- Precisión: Esta métrica nos dice cuántos positivos de los que han sido diagnosticados positivos son de verdad positivos. Una precisión alta es importante para no fallar y tratar diagnósticos positivos que en realidad no lo son. Una precisión baja podría producir exceso de gente en los hospitales y un gasto excesivo en recursos innecesarios.
- Recall: Esta métrica nos dice cuántas personas que deberían ser positivas son positivas. Una recall lo más alta posible es un requisito fundamental para no dejar a ningún paciente sin un tratamiento necesario para él.

La tabla siguiente resume los resultados obtenidos hasta el momento, tanto de Deep Learning como de Machine Learning:

Dataset	Modelo	Técnica de Balanceo	Precision (clase minoritaria)	Recall (clase minoritaria)	Precision (clase mayoritaria)	Recall (clase mayoritaria)
Stroke	Logistic Regression	Random Undersampling	0.21	0.64	0.98	0.92
Stroke	Logistic Regression	Random Oversampling	0.19	0.64	0.97	0.94

Stroke	Logistic Regression	SMOTE	0.20	0.64	0.98	0.94
Stroke	Logistic Regression	Tomek Links	0.22	0.62	0.97	0.96
Stroke	Logistic Regression	SMOTE + Tomek links	0.20	0.64	0.98	0.94
Stroke	Logistic Regression	Sin técnica	0.22	0.58	0.97	0.96
Stroke	Random Forest	Random Undersampling	0.26	0.28	0.98	0.90
Stroke	Random Forest	Random Oversampling	0.14	0.72	0.98	0.82
Stroke	Random Forest	SMOTE	0.14	0.52	0.97	0.90
Stroke	Random Forest	Tomek Links	0.22	0.42	0.98	0.88
Stroke	Random Forest	SMOTE + Tomek links	0.15	0.50	0.98	0.83
Stroke	Random Forest	Sin técnica	0.22	0.40	0.98	0.89
Stroke	XGBoost	Random Undersampling	0.17	0.66	0.98	0.93
Stroke	XGBoost	Random Oversampling	0.19	0.38	0.97	0.96
Stroke	XGBoost	SMOTE	0.12	0.56	0.97	0.84
Stroke	XGBoost	Tomek Links	0.18	0.40	0.98	0.93
Stroke	XGBoost	SMOTE + Tomek links	0.12	0.64	0.98	0.85
Stroke	XGBoost	Sin técnica	0.21	0.46	0.97	0.94
Stroke	MLP	Random Undersampling	0.17	0.60	0.98	0.85
Stroke	MLP	Random Oversampling	0.15	0.48	0.97	0.86
Stroke	MLP	SMOTE	0.12	0.68	0.98	0.75
Stroke	MLP	Tomek Links	0.19	0.36	0.97	0.92
Stroke	MLP	SMOTE + Tomek links	0.15	0.28	0.96	0.92
Stroke	MLP	Sin técnica	0.16	0.44	0.97	0.88
Heart Disease	Logistic Regression	Random Undersampling	0.30	0.57	0.95	0.89
Heart Disease	Logistic Regression	Random Oversampling	0.30	0.56	0.95	0.90
Heart Disease	Logistic Regression	SMOTE	0.30	0.55	0.95	0.89
Heart Disease	Logistic Regression	Tomek Links	0.30	0.56	0.95	0.89

Heart Disease	Logistic Regression	SMOTE + Tomek links	0.30	0.55	0.95	0.89
Heart Disease	Logistic Regression	Sin técnica	0.30	0.56	0.95	0.89
Heart Disease	Random Forest	Random Undersampling	0.32	0.52	0.95	0.89
Heart Disease	Random Forest	Random Oversampling	0.29	0.59	0.95	0.88
Heart Disease	Random Forest	SMOTE	0.31	0.50	0.95	0.90
Heart Disease	Random Forest	Tomek Links	0.31	0.56	0.95	0.89
Heart Disease	Random Forest	SMOTE + Tomek links	0.30	0.52	0.95	0.90
Heart Disease	Random Forest	Sin técnica	0.31	0.57	0.95	0.89
Heart Disease	XGBoost	Random Undersampling	0.31	0.56	0.95	0.89
Heart Disease	XGBoost	Random Oversampling	0.30	0.58	0.95	0.89
Heart Disease	XGBoost	SMOTE	0.28	0.50	0.95	0.90
Heart Disease	XGBoost	Tomek Links	0.30	0.58	0.96	0.88
Heart Disease	XGBoost	SMOTE + Tomek links	0.30	0.47	0.95	0.88
Heart Disease	XGBoost	Sin técnica	0.31	0.56	0.95	0.91
Heart Disease	MLP	Random Undersampling	0.31	0.54	0.95	0.89
Heart Disease	MLP	Random Oversampling	0.31	0.54	0.95	0.89
Heart Disease	MLP	SMOTE	0.32	0.52	0.95	0.90
Heart Disease	MLP	Tomek Links	0.31	0.53	0.95	0.90
Heart Disease	MLP	SMOTE + Tomek links	0.32	0.51	0.95	0.90
Heart Disease	MLP	Sin técnica	0.31	0.55	0.95	0.89

Tabla 1. Resultados de las predicciones según dataset, modelo y técnica de balanceo

En la tabla podemos comprobar cómo, en el caso de la clase mayoritaria, la precisión y la recall superan en su mayoría el 90%. Sin embargo, en la clase minoritaria se observa una caída notable en la precisión, que en los casos más favorables apenas alcanza el 32%. La recall presenta valores algo más elevados, aunque siguen siendo insuficientes, oscilando generalmente entre el 40% y el 60%. La combinación de estas dos métricas, por tanto, nos da unos resultados para nada positivos.

Viendo los resultados, queda más claro aún que las técnicas usadas no han conseguido predecir los resultados de una manera precisa. Los resultados sugieren una gran tendencia a escoger la clase mayoritaria por encima de la minoritaria, acertando en un bajo porcentaje las

personas que de verdad sí tenían la enfermedad o intentando subir este número a costa de añadir gente sana al lado positivo.

2.1. Resultados inflados por configuración incorrecta

Dentro de la creación de código hubo un error crucial que hizo que el entrenamiento, de repente, diera resultados que alertaban de un progreso importante hacia un modelo con un rendimiento muy óptimo. Este error se debió a la introducción de la estrategia de smote antes de hacer el split de train y test, lo que produjo que el train estuviera contaminado con información que luego aparecería en test (o puntos sintéticos derivados). Este tipo de problema es conocido como Data Leakage (o fuga de datos) y es la utilización de datos de la columna target en el entrenamiento. Un ejemplo de resultado obtenido a través de este método es este:

	Precision	**Recall**
No stroke	0.84	0.72
Stroke	0.76	0.87

Tabla 2. Ejemplo de resultados de predicción con Data Leakage

Como se puede observar, las métricas de precisión y recall disminuyen en la clase mayoritaria, mientras que las de la clase minoritaria aumentan considerablemente. Dado que se habla dentro de un contexto clínico, la correcta detección de la clase 1 (casos positivos) resulta prioritaria. En este sentido, se aprecia una mejora aparente de los resultados. Esta mejora, que en un principio puede parecer muy buena, se debe a una fuga de información, por lo que solo representa un salto cualitativo artificial. Este tipo de errores no serían aplicables en un entorno real, pero enseñan como pequeños cambios pueden llevar a grandes diferencias.

3. Conclusiones previas

A pesar de haber aplicado técnicas dedicadas al balanceo de clases, como pueden ser SMOTE o Random UnderSampling, no se han podido observar mejoras consistentes en las predicciones realizadas. Aumentos de métricas como por ejemplo la recall, agravan la situación de la precisión, no habiéndose podido encontrar un equilibrio con el cual poder aumentar las dos métricas. Lo que puede sugerir esto es que el desbalanceo de clases no es el único factor que limita las predicciones y, como línea futura de trabajo, sería recomendable identificar y tratar estos otros factores.

Al final, esto muestra uno de los problemas a resolver del desbalanceo de clases: como mejorar la detección sin aumentar el número de falsos positivos. En contextos clínicos, un porcentaje muy pequeño de falsos positivos puede ser insignificante si a costa de ello se detectan un número mayor de casos, más en el caso de este proyecto el aumento de falsos

positivos no ha sido mínimo, consecuentemente afectando en gran medida al rendimiento general.

El análisis realizado también revela algunos puntos a destacar. El más significativo de ellos es la importancia de no tratar los resultados de forma global, sino desgranarlos comprobando de verdad que son aplicables en el caso real. En el caso de escoger la precisión global, se vería un dato de precisión por encima del 0.9 en muchos de los casos. Esto en ningún caso quiere decir que el modelo haya dado buenos resultados, sino que, al ser la clase mayoritaria de un tamaño muy superior a la minoritaria, esta coge un gran peso en el cálculo de la medida global.

En conclusión, los resultados que se han obtenido han reflejado las dificultades a les cuales hay que enfrentarse en un problema donde los datos tienen un fuerte desbalanceo de clases. Aún haber probado diferentes modelos y técnicas, no se ha podido lograr un rendimiento satisfactorio, pudiéndose pensar que el desbalanceo de clases no es el único problema al que habría que enfrentarse, sino que habría más, como la posible falta de datos o columnas relevantes. Dicho esto, el futuro de este proyecto no debería ya centrarse solo en la mejora de los algoritmos sino en hacer un análisis mucho mas exhaustivo de los datos existentes e intentar añadir más información a la actual, todo esto con el objetivo de fundamentar o refutar de forma sólida la hipótesis de que no existe ningún tipo de patrón en los datos.