

Análisis de datos clínicos

# Estudio sobre indicadores de salud para la diabetes

Práctica final

Guillermo Varela Carbajal y Raquel Gosálbez Sirvent

# Índice

---

I.- NIVEL BÁSICO.....	3
I.1.- Análisis exploratorio del conjunto de datos .....	3
I.2.- Procesamiento de los datos .....	8
I.3.- Primeros esfuerzos de modelización.....	9
I.3.a- Algoritmos de clasificación .....	9
I.3.b.- Validación Cruzada .....	10
I.3.c.- Ajuste de parámetros .....	10
I.3.d.- Equilibrado de clases.....	11
I.3.e.- Contraste de hipótesis.....	14
II.- NIVEL MEDIO .....	14
II.1.- Algoritmos de selección automática de atributos relevantes.....	15
II.2.- Clasificadores.....	15
II.2.a.- Dummy Classifier (librería Scikit-learn) .....	16
II.2.b.- Logistic Regression (librería Scikit-learn).....	16
II.2.c.- Random Forest Classifier (librería Scikit-learn).....	17
II.2.d.- Balanced Random Forest Classifier (librería Imbalanced-learn) .....	18
II.2.e.- Light Gradient Boosting Machine (librería LightGBM) .....	19
II.2.f.- eXtreme Gradient Boosting (librería XGBoost) .....	20
II.2.g.- Modelos descartados: CatBoost y Stacking Ensemble .....	21
II.3.- Ajuste básico de hiperparámetros: Grid Search.....	22
II.4.- Ponderación de clases mediante funciones de coste.....	23
II.5.- Comparación de modelos mediante pruebas estadísticas.....	24
II.6.- Métricas de evaluación de la clasificación binaria .....	25
II.6.a.- Exactitud (Accuracy) .....	25
II.6.b – Precisión (Precision) .....	26
II.6.c.- Sensibilidad (Recall) .....	27
II.6.d.- Puntuación F1 (F1-score).....	28
II.6.e – Área bajo la curva ROC (AUC-ROC) .....	29
III.- NIVEL AVANZADO .....	30
III.1.- Técnicas de reducción de la dimensionalidad: UMAP .....	31
III.2.- Modularización y funciones auxiliares .....	32
III.2.a.- Selección de núcleos para paralelización .....	32
III.2.b.- Estrategia de validación cruzada .....	32
III.2.c.- Cálculo de métricas de evaluación .....	32

III.2.d.- Visualización comparativa de resultados .....	33
III.2.e.- Análisis estadístico de los modelos .....	33
III.2.f.- Optimización del umbral para maximizar el F1-Score .....	33
III.2.g.- Interpretabilidad con SHAP .....	34
III.3.- Buscadores de hiperparámetros avanzados.....	34
III.3.a.- Randomized Search (librería Scikit-Learn).....	34
III.3.b.- Optuna (librería Optuna).....	35
III.4.- Estructura del ajuste de los modelos de clasificación .....	37
III.4.a.- Fase 1: Tuning de los datos .....	37
III.4.b.- Fase 2: Tuning del modelo .....	37
III.4.c.- <i>tuning_data_model()</i> .....	38
III.4.d.- Final model.....	38
III.5.- SHAP .....	38
III.6.- Resultados obtenidos .....	39
III.6.a.- Clasificador Dummy.....	39
III.6.b.- Logistic Regression .....	40
III.6.c.- Random Forest Classifier .....	47
III.6.d.- Balanced Random Forest Classifier .....	52
III.6.e.- LightGBM .....	58
III.6.f.- XGBoost.....	64
III.6.g.- Resumen de modelos finales.....	69
III.7.- Resultado final y conclusiones.....	70
REFERENCIAS.....	71

Con el fin de estructurar la práctica de manera clara y coherente, el desarrollo se ha organizado en tres niveles progresivos: Nivel Básico, Nivel Medio y Nivel Avanzado. Cada uno de estos niveles refleja el grado de profundidad en el tratamiento del problema, tanto en términos de técnicas aplicadas como de análisis realizados.

## I.- NIVEL BÁSICO

---

Este primer bloque se centra en la fase inicial del estudio, en la que se realiza una exploración exhaustiva del conjunto de datos, se establecen las bases del preprocesamiento y se desarrollan los primeros modelos de clasificación. El objetivo principal de este nivel es comprender la estructura y calidad de los datos, identificar posibles retos asociados (como el desbalance de clases o la ausencia de normalidad en las variables numéricas) y diseñar una primera estrategia de modelización que sirva como referencia para futuras mejoras.

Es importante señalar que tanto este Nivel Básico como el Nivel Medio tienen un carácter exploratorio y formativo, por lo que en ellos se exponen pruebas, decisiones metodológicas y análisis preliminares que sirvieron de guía para avanzar hacia soluciones más sofisticadas. Los resultados mostrados aquí no deben considerarse definitivos, sino una etapa intermedia hacia el diseño final que se presenta en el Nivel Avanzado (bloque III), donde se integran las estrategias más eficaces y se consolidan los modelos finales validados.

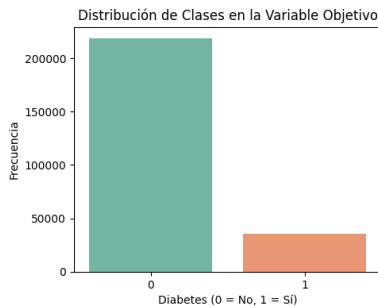
### I.1.- Análisis exploratorio del conjunto de datos

El análisis comenzó con la carga del conjunto de datos, compuesto por más de 250.000 registros y 22 columnas, de las cuales 21 son variables predictoras y una corresponde a la variable objetivo (*Diabetes\_binary*). Las variables predictoras se almacenaron en el dataframe *X*, y la variable diana en *Y*. Para facilitar el análisis conjunto, se unificaron en un solo dataframe llamado *data*.

Una primera inspección visual mediante *data.head()* reveló que las variables estaban codificadas numéricamente y que de momento no existían valores perdidos ni inválidos. Las variables categóricas ya estaban transformadas en formato numérico, y se confirmó que los ceros representaban respuestas válidas, no datos faltantes. Esta codificación uniforme facilitó el preprocesamiento posterior.

Se observó que la mayoría de variables eran categóricas binarias (dicotómicas). Solo tres eran numéricas (*BMI*, *MentHlth*, *PhysHlth*), y cuatro eran categóricas ordinales (*GenHlth*, *Age*, *Education* e *Income*).

Se utilizó la librería *ydata\_profiling* (función *ProfileReport()*) para generar un informe exploratorio completo. Este análisis confirmó la ausencia de valores nulos y destacó un porcentaje de duplicados del 4,5% (más de 10.000 filas). Sin embargo, tras consultar la fuente oficial del dataset (UCI Machine Learning Repository. (n.d.). *CDC Diabetes Health Indicators Data Set*), se concluyó que estas coincidencias no representaban duplicación real, sino repeticiones legítimas debidas al reducido número de combinaciones posibles entre variables principalmente binarias.

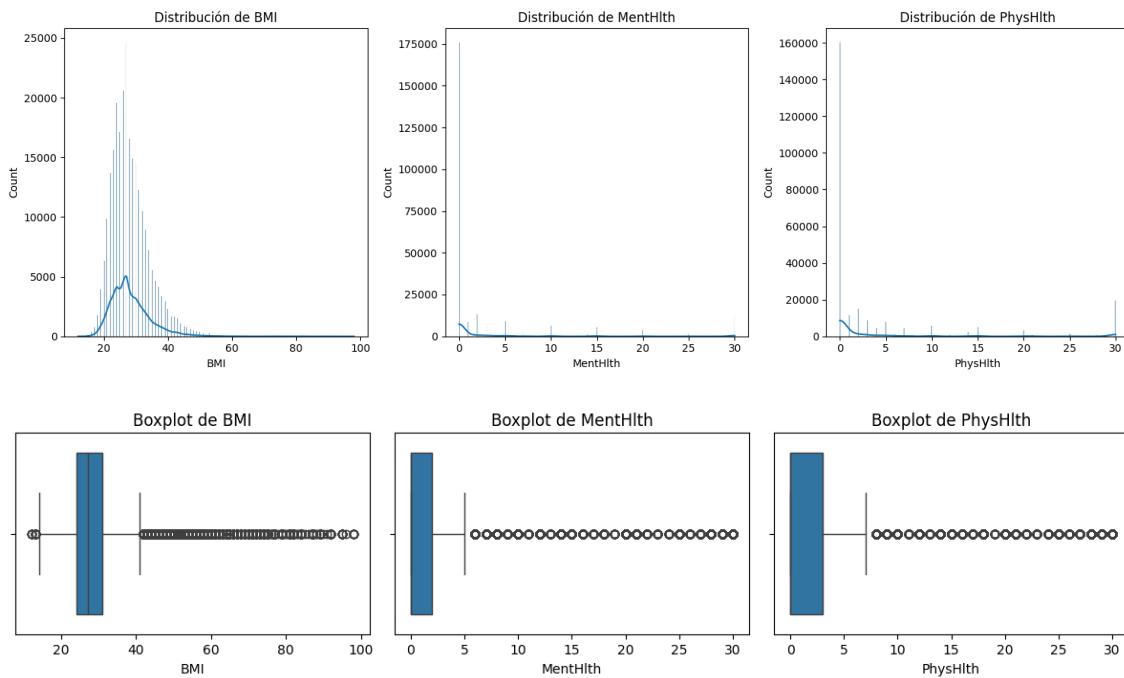


La visualización de las distribuciones mostró desequilibrios relevantes en varias variables binarias. Por ejemplo, las categorías positivas en CholCheck (77.0%), Stroke (75.5%), HeartDiseaseorAttack (55.0%), HvyAlcoholConsump (68.8%), AnyHealthcare (71.8%) y NoDocbcCost (58.3%), además de la propia variable objetivo, Diabetes\_binary (86,1%). Esta desproporción anticipa un reto importante en la fase de modelado.

#### Alerts

Dataset has 11369 (4.5%) <a href="#">duplicate rows</a>	<a href="#">Duplicates</a>
<a href="#">CholCheck</a> is highly imbalanced (77.0%)	<a href="#">Imbalance</a>
<a href="#">Stroke</a> is highly imbalanced (75.5%)	<a href="#">Imbalance</a>
<a href="#">HeartDiseaseorAttack</a> is highly imbalanced (55.0%)	<a href="#">Imbalance</a>
<a href="#">HvyAlcoholConsump</a> is highly imbalanced (68.8%)	<a href="#">Imbalance</a>
<a href="#">AnyHealthcare</a> is highly imbalanced (71.8%)	<a href="#">Imbalance</a>
<a href="#">NoDocbcCost</a> is highly imbalanced (58.3%)	<a href="#">Imbalance</a>

Respecto a las variables numéricas (BMI, PhysHlth, MentHlth), se estudiaron sus distribuciones con histogramas y si presentaban outliers con diagramas de caja.



Para comprobar su normalidad se aplicaron tres pruebas estadísticas:

- Shapiro-Wilk: adecuada para muestras pequeñas, evalúa desviaciones de normalidad.
- D'Agostino K<sup>2</sup>: evalúa simetría y curtosis de la distribución.
- Anderson-Darling: similar a Shapiro pero con mayor sensibilidad en los extremos.

```

--- Análisis de normalidad para BMI ---
Shapiro-Wilk Test: Estadístico=0.8717, p=0.0000 (Rechaza H0)
D'Agostino's K^2 Test: Estadístico=126255.7185, p=0.0000 (Rechaza H0)
Anderson-Darling Test:
    Nivel de significancia 15.0% - Valor crítico: 0.5760 (Rechaza H0)
    Nivel de significancia 10.0% - Valor crítico: 0.6560 (Rechaza H0)
    Nivel de significancia 5.0% - Valor crítico: 0.7870 (Rechaza H0)
    Nivel de significancia 2.5% - Valor crítico: 0.9180 (Rechaza H0)
    Nivel de significancia 1.0% - Valor crítico: 1.0920 (Rechaza H0)

--- Análisis de normalidad para MntHlth ---
Shapiro-Wilk Test: Estadístico=0.4869, p=0.0000 (Rechaza H0)
/usr/local/lib/python3.11/dist-packages/scipy/stats/_axis_nan_policy.py:573:
    res = hypothesis_fun_out(*samples, **kwdss)
D'Agostino's K^2 Test: Estadístico=137190.8598, p=0.0000 (Rechaza H0)
Anderson-Darling Test:
    Nivel de significancia 15.0% - Valor crítico: 0.5760 (Rechaza H0)
    Nivel de significancia 10.0% - Valor crítico: 0.6560 (Rechaza H0)
    Nivel de significancia 5.0% - Valor crítico: 0.7870 (Rechaza H0)
    Nivel de significancia 2.5% - Valor crítico: 0.9180 (Rechaza H0)
    Nivel de significancia 1.0% - Valor crítico: 1.0920 (Rechaza H0)

--- Análisis de normalidad para PhysHlth ---
Shapiro-Wilk Test: Estadístico=0.5385, p=0.0000 (Rechaza H0)
D'Agostino's K^2 Test: Estadístico=104008.5163, p=0.0000 (Rechaza H0)
Anderson-Darling Test:
    Nivel de significancia 15.0% - Valor crítico: 0.5760 (Rechaza H0)
    Nivel de significancia 10.0% - Valor crítico: 0.6560 (Rechaza H0)
    Nivel de significancia 5.0% - Valor crítico: 0.7870 (Rechaza H0)
    Nivel de significancia 2.5% - Valor crítico: 0.9180 (Rechaza H0)
    Nivel de significancia 1.0% - Valor crítico: 1.0920 (Rechaza H0)

```

por realizar un análisis bivariado personalizado, adaptando las técnicas estadísticas al tipo de variables implicadas.

Cuando ambas variables eran cuantitativas, se utilizó el coeficiente de correlación de Spearman. A diferencia de Pearson, que mide la relación lineal entre dos variables continuas y sensibles a valores extremos, Spearman evalúa relaciones monótonas (crecientes o decrecientes, pero no necesariamente lineales) y se basa en los rangos de los valores. Esto lo convierte en una opción más robusta para datos no normales o con presencia de outliers, como los que presentaba este conjunto de datos.

Cuando ambas variables eran categóricas, se empleó la V de Cramer. Esta medida, basada en el estadístico chi-cuadrado, cuantifica la fuerza de asociación entre dos variables categóricas, sin asumir orden entre las categorías. Su valor oscila entre 0 (sin asociación) y 1 (asociación perfecta), lo que permite interpretar la intensidad de la relación detectada.

En los casos donde se relacionaba una variable cuantitativa con una categórica, se emplearon pruebas no paramétricas en función del número de categorías:

- Si la variable categórica tenía dos niveles (por ejemplo, Diabetes\_binary), se aplicó la prueba U de Mann-Whitney, que compara las distribuciones de la variable cuantitativa entre los dos grupos. Además, se calculó el tamaño del efecto mediante el coeficiente  $r = |z| / \sqrt{n}$ , que indica la magnitud de la diferencia observada.
- Si la variable categórica tenía más de dos niveles, se utilizó la prueba de Kruskal-Wallis, equivalente no paramétrica del ANOVA. Para cuantificar la fuerza de la relación, se empleó el estadístico  $\eta^2$  (eta cuadrado), que estima el porcentaje de la varianza explicada por el factor categórico.

Todos los resultados de este análisis se sintetizaron en una matriz de comparaciones visualizada como un mapa de calor, donde cada celda representa la intensidad de la asociación entre un par de variables según la métrica correspondiente y el color de la misma la prueba que se ha realizado. Esta representación permitió identificar de forma clara los vínculos más relevantes del conjunto de datos para su posterior análisis predictivo.

Todas las pruebas coincidieron en rechazar la hipótesis de normalidad. Debido a esto, se optó por aplicar pruebas estadísticas no paramétricas en el análisis bivariado.

Aunque la función ProfileReport() ofrece una matriz de correlaciones e interacciones entre variables, su análisis se basa por defecto en el coeficiente de correlación de Pearson, que asume normalidad y relaciones lineales entre las variables. Esta suposición no se cumplía en nuestro caso, ya que las variables cuantitativas presentaban distribuciones claramente no normales. Por ello, se optó

		Matriz de Análisis																				
		Tipo de Análisis																				
		Spearman																				
		Mann-Whitney U																				
		Kruskal-Wallis																				
		Cramer's V																				
		Sext																				
Gentith		0.08	0.15	0.20	0.30	0.21	0.05	0.16	0.19	0.27	0.27	0.11	0.12	0.04	0.05	0.17	0.50	0.02	0.07	0.07	0.27	
Age		0.08		0.07	0.10	0.35	0.29	0.10	0.14	0.13	0.23	0.10	0.09	0.03	0.04	0.15	0.13	0.21	0.04	0.02	0.03	0.00
Education		0.15	0.07		0.22	0.14	0.07	0.01	0.18	0.08	0.10	0.20	0.12	0.16	0.03	0.13	0.11	0.20	0.04	0.02	0.00	0.02
Income		0.20	0.10	0.22		0.18	0.09	0.02	0.13	0.13	0.14	0.20	0.08	0.15	0.06	0.16	0.20	0.32	0.13	0.01	0.03	0.05
HighBP		0.30	0.35	0.14	0.18		0.30	0.10	0.10	0.13	0.21	0.13	0.04	0.06	0.00	0.04	0.02	0.22	0.05	0.24	0.02	0.13
HighChol		0.21	0.29	0.07	0.09	0.30		0.09	0.09	0.18	0.08	0.04	0.04	0.01	0.04	0.01	0.14	0.03	0.14	0.04	0.10	
CholCheck		0.05	0.10	0.01	0.02	0.10	0.09		0.01	0.02	0.04	0.00	0.02	0.01	0.02	0.12	0.06	0.04	0.02	0.04	0.01	0.03
Smoker		0.16	0.14	0.18	0.13	0.10	0.09	0.01		0.06	0.11	0.09	0.08	0.03	0.10	0.02	0.05	0.12	0.09	0.02	0.05	0.08
Stroke		0.19	0.13	0.08	0.13	0.13	0.09	0.02	0.06		0.20	0.07	0.01	0.04	0.02	0.01	0.03	0.18	0.00	0.02	0.04	0.11
HeartDiseasesAttack		0.27	0.23	0.10	0.14	0.21	0.19	0.04	0.11	0.20		0.09	0.02	0.04	0.03	0.02	0.03	0.21	0.09	0.06	0.03	0.14
PhysActivity		0.27	0.10	0.20	0.20	0.13	0.08	0.00	0.09	0.07	0.09		0.14	0.15	0.01	0.04	0.06	0.25	0.03	0.14	0.07	0.16
Fruits		0.11	0.09	0.12	0.08	0.04	0.04	0.02	0.08	0.01	0.02	0.14		0.25	0.04	0.03	0.04	0.05	0.09	0.10	0.05	0.04
Veggies		0.12	0.03	0.16	0.15	0.06	0.04	0.01	0.03	0.04	0.04	0.15	0.25		0.02	0.03	0.03	0.08	0.06	0.07	0.03	0.05
HeavyAlcoholConsump		0.04	0.04	0.03	0.06	0.00	0.01	0.02	0.10	0.02	0.03	0.01	0.04	0.02		0.01	0.00	0.04	0.01	0.05	0.03	0.02
AnyHealthcare		0.05	0.15	0.13	0.16	0.04	0.04	0.12	0.02	0.01	0.02	0.04	0.03	0.03	0.01		0.23	0.01	0.02	0.01	0.04	0.01
NoDrogsCost		0.17	0.13	0.11	0.20	0.02	0.01	0.06	0.05	0.03	0.03	0.06	0.04	0.03	0.00	0.23		0.12	0.04	0.05	0.15	0.13
DifWalk		0.50	0.21	0.20	0.32	0.22	0.14	0.04	0.12	0.18	0.21	0.25	0.05	0.08	0.04	0.01	0.12		0.07	0.18	0.15	0.36
Sex		0.02	0.04	0.04	0.13	0.05	0.03	0.02	0.09	0.06	0.09	0.03	0.09	0.06	0.01	0.02	0.04	0.07		0.09	0.10	0.06
BMI		0.07	0.02	0.02	0.01	0.24	0.14	0.04	0.02	0.02	0.06	0.14	0.10	0.07	0.05	0.01	0.05	0.18		0.05	0.12	
MentalHlth		0.07	0.03	0.00	0.03	0.02	0.04	0.01	0.05	0.04	0.03	0.07	0.05	0.03	0.04	0.15	0.15	0.10	0.05		0.31	
PhysicalHlth		0.27	0.00	0.02	0.05	0.13	0.10	0.03	0.08	0.11	0.14	0.16	0.04	0.05	0.02	0.01	0.13	0.36	0.06	0.12	0.31	

Los resultados del análisis bivariado se interpretaron utilizando umbrales estandarizados para cada tipo de prueba, con el objetivo de categorizar la magnitud de las relaciones detectadas. A continuación, se presentan los criterios empleados, basados en la literatura estadística y adaptados al contexto del análisis exploratorio de datos de salud:

#### Correlación de Spearman:

- < 0.10: correlación despreciable
- 0.10–0.30: correlación débil
- 0.30–0.50: correlación moderada
- 0.50–0.70: correlación fuerte
- ≥ 0.70: correlación muy fuerte

#### Mann-Whitney U:

- ≈ 0.10: efecto pequeño
- ≈ 0.30: efecto moderado
- ≈ 0.50: efecto grande

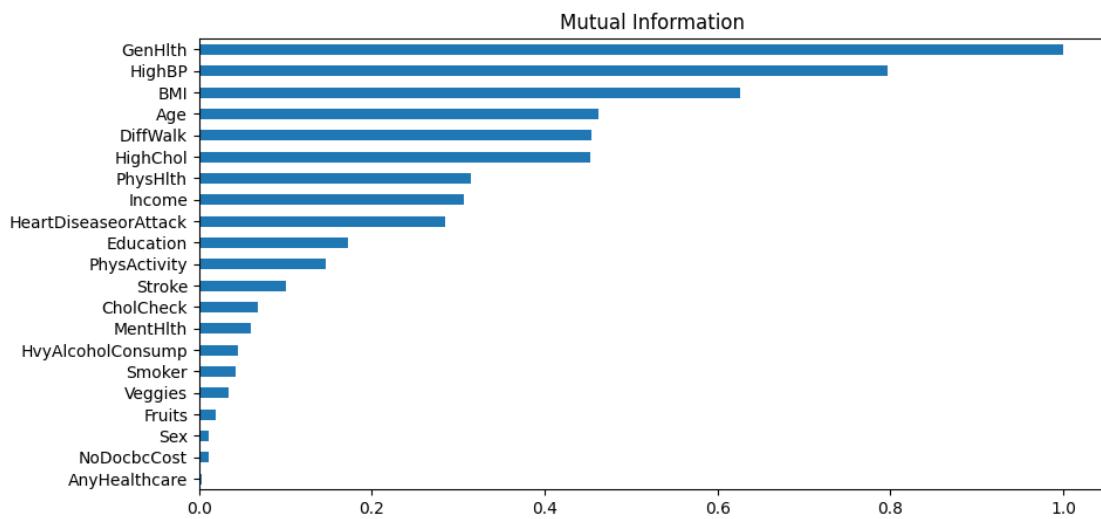
#### Kruskal-Wallis:

- 0.01–0.06: efecto pequeño
- 0.06–0.14: efecto moderado
- ≥ 0.14: efecto grande

#### V de Cramer:

- < 0.10: asociación despreciable
- 0.10–0.30: asociación débil
- 0.30–0.50: asociación moderada
- $\geq 0.50$ : asociación fuerte

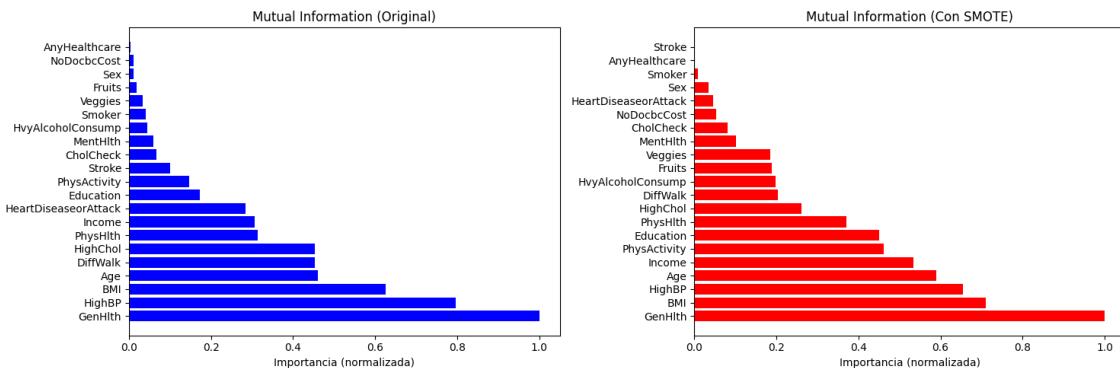
Siguiendo con la exploración de los datos, también se empleó el cálculo de información mutua (mutual information) para estimar qué variables predictoras proporcionan más información sobre la variable objetivo (Diabetes\_binary). Esta métrica, de origen teórico en la teoría de la información, mide la reducción de incertidumbre sobre una variable al conocer otra. A diferencia de la correlación lineal, la información mutua detecta cualquier tipo de dependencia, incluso no lineal, lo que la convierte en una herramienta muy útil en contextos con variables categóricas o no distribuidas normalmente.



En esta primera aproximación, se observó que algunas de las variables con menor puntuación de información mutua eran aquellas con una distribución muy desbalanceada entre clases (AnyHealthcare, NoDocbcCost). Esto podía sugerir que el desequilibrio en la representación de clases podría estar influyendo en el resultado del análisis.

Para comprobarlo, se aplicó la técnica de SMOTE (Synthetic Minority Over-sampling Technique), una estrategia de *oversampling* que genera nuevas instancias sintéticas de la clase minoritaria. Su uso en el conjunto de entrenamiento permite mitigar el sesgo hacia la clase mayoritaria sin eliminar información.

Tras equilibrar la muestra con SMOTE, se repitió el cálculo de información mutua, lo que permitió observar ciertos cambios en el ranking de variables más relevantes, confirmando que el desequilibrio original tenía un efecto en los resultados. Aun así, la variable GenHlth mantuvo su posición destacada como el predictor más informativo.



## 1.2.- Procesamiento de los datos

Como resultado del análisis exploratorio, se procedió a una primera reducción de variables basada en dos criterios complementarios: baja relevancia individual y redundancia informativa.

En primer lugar, se eliminaron las cinco variables con menor información mutua respecto a la variable objetivo (Diabetes\_binary): AnyHealthcare, Sex, Smoker, Stroke y NoDocbcCost. Estas variables aportaban muy poca información predictiva, lo que las hacía prescindibles en esta primera aproximación.

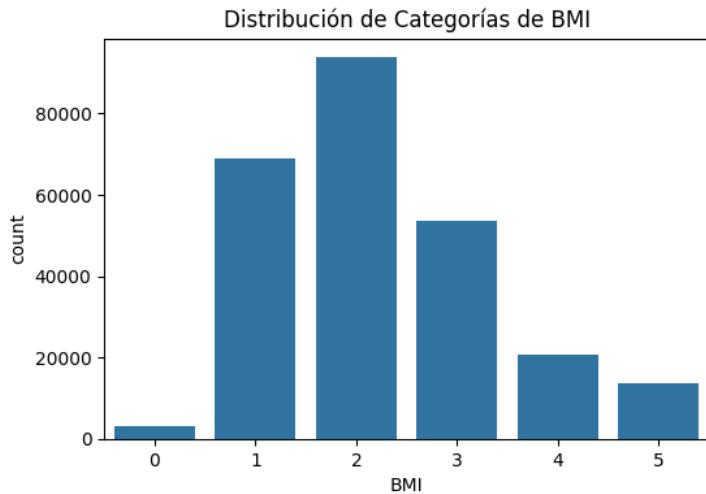
En segundo lugar, se identificó una fuerte correlación entre las variables DiffWalk y PhysHlth con la variable GenHlth, que fue la más informativa según el análisis anterior. Dado que GenHlth sintetiza de forma subjetiva el estado general de salud, se consideró que las otras dos variables estaban aportando información redundante, por lo que también fueron descartadas.

La exclusión de estas siete variables dio lugar a una nueva versión del conjunto de datos, representada por los dataframes *X\_filtered* (variables predictoras filtradas) e *Y\_filtered* (variable objetivo, realmente no filtrada pero se le confirió ese nombre para mejor legibilidad del código). Esta selección reducida de variables servirá como una de las configuraciones base para comparar el rendimiento de los modelos (“Configuración 1: variables seleccionadas manualmente”).

Posteriormente, se quiso tratar los outliers excesivos y con valores anormales que presentaba la variable BMI. Se barajaron distintas formas de abordar la cuestión: eliminando valores, normalizarlas, no tratarlo, etc. Finalmente se decidió por categorizar la variable en seis categorías utilizadas muy frecuentemente:

- (0) Infrapeso: por debajo de 18.5
- (1) Normopeso: 18.5–24.9
- (2) Pre-obesidad: 25.0–29.9
- (3) Obesidad clase I: 30.0–34.9
- (4) Obesidad clase II: 35.0–39.9
- (5) Obesidad clase III: por encima de 40

Solucionado el problema de los valores extremos, la nueva variable BMI categorizada (tanto en *X* como en *X\_filtered*) presenta la siguiente distribución.



Por último, se realizó una división estratificada de los datos (*stratified train-test split*), separando un 80% de los registros para el conjunto de entrenamiento y un 20% para el conjunto de prueba, creando así  $X_{train}$ ,  $y_{train}$ ,  $X_{test}$  e  $y_{test}$  (y sus homólogos *\_filtered*). La estratificación consiste en preservar la proporción original de clases de la variable objetivo (0 y 1) en ambos subconjuntos, lo cual es especialmente importante en datasets desbalanceados como el presente, donde la clase minoritaria representa solo el 13,9% del total.

Para asegurar la reproducibilidad de los resultados, se fijó una semilla (`random_state=42`) en todas las funciones que implicaban aleatoriedad, como la partición de los datos o la aplicación de técnicas de sobremuestreo. Esto permite que cualquier persona que ejecute el mismo código obtenga exactamente los mismos resultados, lo cual es un principio esencial en ciencia de datos.

### I.3.- Primeros esfuerzos de modelización

Este apartado recoge la primera aproximación al problema de clasificación, diseñada en línea con los criterios establecidos en la rúbrica de evaluación. Su propósito es mostrar el punto de partida metodológico del estudio, así como documentar la evolución progresiva hacia estrategias más avanzadas.

Si bien las técnicas empleadas en esta fase inicial fueron posteriormente mejoradas y ampliadas en los bloques correspondientes al Nivel Medio (bloque II) y Nivel Avanzado (bloque III), se considera relevante exponer aquí los primeros enfoques implementados, tanto por su valor formativo como por su utilidad para contextualizar los resultados finales.

#### I.3.a- Algoritmos de clasificación

Como primer paso en la fase de modelización, se implementaron tres clasificadores básicos utilizando la biblioteca scikit-learn, con el objetivo de establecer una línea base de comparación y explorar cómo distintos algoritmos respondían ante el conjunto de datos.

El primer modelo fue un `DummyClassifier`, que predice siempre la clase mayoritaria. Este modelo no utiliza información de las variables predictoras y actúa como referencia mínima para evaluar la utilidad de los modelos posteriores.

A continuación, se entrenaron dos clasificadores reales:

- Una regresión logística, seleccionada por su simplicidad y carácter interpretativo.

- Un Random Forest, como ejemplo de modelo basado en ensamblado de árboles de decisión, capaz de capturar interacciones no lineales entre variables.

Estos tres modelos proporcionaron una primera aproximación al problema y permitieron detectar limitaciones iniciales, como el impacto del desequilibrio de clases o la necesidad de ajustar hiperparámetros. En etapas posteriores del proyecto (descritas en los niveles medio y avanzado) se incorporaron modelos más sofisticados como `BalancedRandomForestClassifier`, `LightGBM` y `XGBoost`.

### I.3.b.- Validación Cruzada

Para evaluar de forma robusta el rendimiento de los modelos de clasificación, se utilizó la técnica de validación cruzada estratificada con 10 particiones (10-CV), implementada mediante la clase `StratifiedKFold` de scikit-learn.

Este método divide el conjunto de datos en 10 subconjuntos de igual tamaño, asegurando que en cada uno se mantenga la misma proporción de clases que en el conjunto original. Esta característica resulta especialmente importante en este proyecto, dado el desequilibrio de clases en la variable objetivo, donde la mayoría de los casos corresponden a personas saludables.

Además, se activó la opción `shuffle=True` para barajar las muestras antes de la partición, reduciendo el riesgo de sesgos relacionados con el orden en que se presentaban los datos. También se fijó el parámetro `random_state=42` para asegurar la reproducibilidad de los resultados.

Este enfoque permite evaluar el comportamiento del modelo en distintas combinaciones de entrenamiento y prueba, reduciendo la dependencia de una única partición. Las métricas obtenidas (en este caso, el AUC) se calcularon en cada fold y se promediaron para obtener una estimación global y más fiable del rendimiento del modelo.

### I.3.c.- Ajuste de parámetros

En esta fase inicial del proyecto, se llevó a cabo un ajuste manual de hiperparámetros con el objetivo de mejorar el rendimiento de los clasificadores en términos de AUC (área bajo la curva ROC), la métrica principal elegida para evaluar la capacidad discriminativa de los modelos. Esta métrica es especialmente adecuada en contextos con clases desbalanceadas, ya que no depende del umbral de clasificación y refleja la calidad del modelo en distinguir entre casos positivos y negativos.

El ajuste consistió en modificar de forma controlada algunos hiperparámetros clave, como `max_depth`, `n_estimators` o `learning_rate`, en modelos de tipo árbol y boosting, observando el impacto de cada combinación en el AUC obtenido mediante validación cruzada. Aunque este procedimiento no fue exhaustivo ni automatizado, resultó útil para familiarizarse con la sensibilidad de los modelos a diferentes configuraciones.

El enfoque empleado se basó en la técnica de ensayo y error, lo que permitió acotar rangos razonables para futuras optimizaciones más sistemáticas. Estos primeros experimentos también proporcionaron valores de referencia con los que comparar los resultados de modelos mejor ajustados en etapas posteriores del estudio.

En fases avanzadas del trabajo, este proceso manual fue reemplazado por métodos automáticos de búsqueda de hiperparámetros, como `GridSearchCV`, `RandomizedSearchCV` y `Optuna`, descritos en detalle en la sección correspondiente al Nivel Avanzado.

### I.3.d.- Equilibrado de clases

En problemas de clasificación binaria como el presente —donde la clase positiva (diabetes/prediabetes) representa solo el 13,9% de los casos— es habitual que los modelos tiendan a favorecer la clase mayoritaria. Esto puede traducirse en altos niveles de exactitud aparente, pero con un rendimiento muy deficiente en la detección de los casos positivos, que son precisamente los más relevantes desde el punto de vista clínico y predictivo.

Para mitigar este efecto, se aplicaron técnicas de balanceo de clases en el conjunto de entrenamiento. Concretamente, se exploraron dos estrategias:

- **Oversampling (sobremuestreo):** consiste en aumentar artificialmente la cantidad de ejemplos de la clase minoritaria, ya sea duplicando registros existentes o generando ejemplos sintéticos (como en el caso de SMOTE).
- **Undersampling (submuestreo):** implica reducir la cantidad de ejemplos de la clase mayoritaria, seleccionando una muestra representativa más pequeña.

Estas técnicas se aplicaron de forma independiente al modelo, es decir, el balanceo se realiza antes de ajustar el clasificador, por lo que cualquier modelo (logístico, árboles, boosting) puede entrenarse sobre datos equilibrados.

Para gestionar este proceso de forma correcta y reproducible, se utilizó la clase `ImbPipeline` de la biblioteca `imblearn`, que permite encadenar pasos de preprocesamiento y remuestreo respetando la lógica del entrenamiento. A diferencia de un Pipeline estándar de `scikit-learn`, `ImbPipeline` está diseñado específicamente para integrarse con transformaciones que modifican el tamaño del conjunto de datos, como `RandomUnderSampler`.

Este comportamiento se gestiona internamente mediante el método `fit_resample()` de los objetos de remuestreo. `ImbPipeline` detecta automáticamente que `RandomUnderSampler` (o `SMOTE`) es una etapa de remuestreo, y por tanto:

- Aplica el balanceo solo durante el entrenamiento, dentro del método `fit()`.
- Omite esa etapa durante la predicción, de modo que los métodos `predict()`, `transform()` o `score()` actúan sobre los datos originales de validación o prueba, sin alterarlos.

Este enfoque garantiza que el modelo no vea datos artificiales fuera del conjunto de entrenamiento, respetando las buenas prácticas metodológicas en machine learning.

#### *Sobremuestreo (oversampling)*

El sobremuestreo es una técnica empleada para corregir el desequilibrio entre clases en problemas de clasificación binaria, como el que aborda este estudio: predecir la presencia de diabetes a partir de indicadores de salud. En este tipo de problemas, es habitual que la clase positiva (pacientes con diabetes o prediabetes) esté subrepresentada frente a la clase negativa (personas saludables). Este desbalance puede provocar que el modelo tienda a predecir siempre la clase mayoritaria, logrando alta exactitud aparente, pero con un desempeño muy pobre en la detección de casos positivos.

El sobremuestreo actúa aumentando la presencia de la clase minoritaria en el conjunto de entrenamiento. Esto puede hacerse de forma simple, duplicando aleatoriamente registros existentes, o de forma más elaborada, generando instancias sintéticas que simulan nuevos casos realistas. Un ejemplo de esto último es `SMOTE` (*Synthetic Minority Over-sampling Technique*), que crea nuevos ejemplos interpolando entre vecinos cercanos dentro del espacio de características.

En este estudio, se implementaron ambas estrategias. La técnica se integró mediante el uso de ImbPipeline (de la librería imblearn), que permite combinar transformaciones de preprocesamiento con técnicas de remuestreo. Es importante destacar que este pipeline aplica el sobremuestreo únicamente durante el entrenamiento del modelo. Esto se debe a que ImbPipeline reconoce que clases como RandomUnderSampler o SMOTE no implementan *transform()*, sino *fit\_resample()*, y por tanto se saltan automáticamente en las fases de predicción y validación, evitando la introducción de datos artificiales en el cálculo de métricas.

En términos cuantitativos, el dataset original contenía 253.680 registros, de los cuales solo 13,93% correspondían a personas con diabetes o prediabetes, es decir, aproximadamente 35.340 casos positivos, frente a 218.340 casos negativos (saludables). Al aplicar sobremuestreo aleatorio, se duplicaron instancias de la clase minoritaria hasta igualar su número al de la clase mayoritaria. Esto implicó crear aproximadamente 183.000 nuevas instancias positivas, alcanzando una proporción 1:1 en el conjunto de entrenamiento. Alternativamente, se aplicó SMOTE, que generó observaciones sintéticas adicionales a partir de los casos reales. El resultado fue un conjunto de entrenamiento con cerca de 436.680 observaciones balanceadas, en el que ambas clases estaban igualmente representadas. Esta transformación forzó al modelo a prestar más atención a los patrones de la clase minoritaria, lo que mejoró su capacidad de detección.

El sobremuestreo se probó con diferentes clasificadores, como regresión logística, Random Forest y XGBoost. En modelos lineales como la regresión logística, el algoritmo ajusta sus coeficientes tras haber visto muchas más instancias positivas, lo que en la práctica aumenta la sensibilidad hacia la clase minoritaria, sin necesidad de modificar la función de coste. En modelos basados en árboles, como Random Forest, el equilibrio se refleja en las muestras de entrenamiento (*bootstrap*) utilizadas por cada árbol, lo que reduce el sesgo hacia la clase mayoritaria.

En el caso de modelos de boosting como LightGBM o XGBoost, aunque es posible aplicar sobremuestreo, en la práctica es más habitual utilizar el parámetro *scale\_pos\_weight* para ajustar el coste de clasificación de la clase minoritaria. Esto permite aprovechar las capacidades internas del algoritmo para gestionar el desbalance sin duplicar físicamente los datos.

El uso del sobremuestreo presentó ventajas claras en cuanto a la capacidad del modelo para detectar pacientes con diabetes, mejorando métricas como el recall o el AUC. Esta técnica permite preservar todos los casos positivos, que en contextos médicos como este suelen ser valiosos y escasos, y maximiza la información que se extrae de ellos.

Sin embargo, cuando se emplea sobremuestreo deben tenerse presentes algunos riesgos:

- El sobreajuste es más probable cuando se duplican instancias reales, ya que el modelo puede memorizar patrones específicos en lugar de generalizar.
- En el caso de SMOTE, es posible que algunas combinaciones de atributos generadas puedan carecer de sentido clínico, introduciendo ruido en el entrenamiento.
- El aumento del tamaño del conjunto de datos implica un mayor coste computacional, especialmente en modelos avanzados.

Este último aspecto fue especialmente determinante en el desarrollo del proyecto, ya que algunos modelos como BalancedRandomForest vieron incrementado su tiempo de entrenamiento de forma tan drástica que su ejecución se volvió inviable.

De esta forma, a pesar de sus beneficios teóricos, el sobremuestreo se abandonó en fases posteriores del estudio debido a sus limitaciones prácticas de escalabilidad. Se optó entonces por técnicas

alternativas más eficientes, como el submuestreo y el ajuste de pesos de clase (véase apartado II.4. del Nivel Medio), que permitieron mantener la capacidad predictiva sin penalizar el rendimiento.

### *Submuestreo (undersampling)*

El submuestreo es la técnica opuesta al sobremuestreo: en lugar de aumentar los ejemplos de la clase minoritaria, reduce el número de instancias de la clase mayoritaria hasta lograr un equilibrio. En el contexto de este estudio, donde las personas sin diagnóstico de diabetes representan la gran mayoría, el submuestreo consistió en seleccionar aleatoriamente un subconjunto de individuos saludables para entrenar el modelo junto a todos los casos positivos (diabéticos o prediabéticos), eliminando el exceso de registros mayoritarios.

En este estudio, el conjunto original incluía 253.680 registros, de los cuales 35.340 pertenecen a personas con diagnóstico de diabetes o prediabetes (13,93%) y 218.340 a personas saludables (86,07%). El submuestreo aleatorio consistió en seleccionar una muestra aleatoria de 35.340 personas saludables para igualar el número de casos positivos. De este modo, el conjunto de entrenamiento resultante quedó balanceado en una proporción 1:1, con un total de 70.680 observaciones. Esta técnica no altera los datos existentes ni introduce instancias sintéticas, lo que reduce el riesgo de generar ruido artificial.

Aunque el submuestreo aleatorio es sencillo de aplicar, también existen variantes más sofisticadas que seleccionan qué ejemplos mayoritarios eliminar en función de su relevancia para la tarea de clasificación:

- NearMiss elimina instancias de la clase mayoritaria que están lejos de los positivos.
- Tomek Links identifica pares de instancias (una de cada clase) que están cerca entre sí y elimina la más redundante.
- Cluster Centroids agrupa ejemplos mayoritarios en clústeres y utiliza únicamente sus centroides, conservando representatividad.

Estas estrategias buscan minimizar la pérdida de información al reducir la clase mayoritaria.

En este proyecto, el submuestreo se aplicó antes del entrenamiento de los modelos mediante ImbPipeline. Esto permitió trabajar con conjuntos de datos equilibrados sin necesidad de modificar la lógica interna del modelo.

En el caso de Random Forest, existen también mecanismos automáticos de balanceo interno, como `class_weight='balanced_subsample'`, que ajusta los pesos en función de la distribución de clases en cada muestra bootstrap. Esta opción permite que cada árbol del bosque entrene con un subconjunto diferente de ejemplos, preservando diversidad y reduciendo el sesgo hacia la clase mayoritaria. Otro enfoque más específico es el uso de un Balanced Random Forest, en el que cada árbol se entrena explícitamente con todos los positivos y un submuestreo igual de los negativos. Este método se explorará en detalle en el Nivel Medio (apartado II.2.d).

En algoritmos de boosting como LightGBM o XGBoost, no existe un submuestreo estratificado nativo por clase. Aunque ambos permiten submuestrear filas o columnas de forma aleatoria (`subsample`, `colsample_bytree`), no están orientados directamente al equilibrio entre clases. Sin embargo, es posible preprocesar los datos aplicando un submuestreo aleatorio de la clase mayoritaria, como se hizo en esta fase del estudio, o utilizar parámetros como `scale_pos_weight` para penalizar el error sobre la clase minoritaria sin eliminar datos, como se llevará a cabo en el Nivel Medio (apartado II.4).

Una de las principales ventajas del submuestreo en este estudio fue su eficiencia computacional. Al reducir el tamaño del conjunto de entrenamiento, se consiguió acelerar significativamente el entrenamiento de los modelos, especialmente con algoritmos exigentes como BalancedRandomForest. Esta reducción de complejidad resultó crítica para la viabilidad del estudio, permitiendo realizar múltiples experimentos con tiempos de ejecución razonables.

Sin embargo, el submuestreo también presenta algunos riesgos:

- Al eliminar parte de la clase mayoritaria, se corre el riesgo de perder ejemplos valiosos que representan subpoblaciones relevantes (por ejemplo, personas sanas de cierta edad o hábitos).
- Puede aumentar la varianza del modelo, ya que diferentes selecciones aleatorias de ejemplos pueden producir modelos con comportamientos distintos. Esta sensibilidad se hizo evidente durante el estudio y se abordó en el Nivel Medio mediante ensambles y repeticiones del muestreo.
- Si se eliminan demasiados ejemplos negativos, el modelo puede dejar de reconocer correctamente los casos sanos atípicos, aumentando los falsos positivos.

Por estas razones, el submuestreo debe aplicarse con criterio y moderación, buscando un equilibrio entre la mejora en la detección de la clase minoritaria y la preservación de la diversidad de la clase mayoritaria. En fases posteriores del estudio se exploraron combinaciones de esta técnica con ajustes de métricas como la F1-score, que ponderan de forma equilibrada precisión y sensibilidad (apartado III.2).

### I.3.e.- Contraste de hipótesis

Como parte del análisis del Nivel Básico, se aplicó un test de contraste de hipótesis no paramétrico para comparar el rendimiento de dos modelos de clasificación sobre el conjunto de datos. El test utilizado fue el Wilcoxon signed-rank test, una prueba adecuada para comparar muestras emparejadas cuando no se puede asumir normalidad en las diferencias, como es el caso habitual en métricas derivadas de validación cruzada.

Concretamente, se compararon los valores de AUC obtenidos por dos clasificadores distintos a lo largo de los mismos 10 subconjuntos de validación generados por validación cruzada estratificada. Esta prueba evalúa si las diferencias observadas en el rendimiento son consistentes y estadísticamente significativas, es decir, si uno de los modelos supera sistemáticamente al otro, más allá de la variabilidad aleatoria entre particiones.

La elección de una prueba no paramétrica responde al hecho de que las diferencias de AUC entre modelos no seguían una distribución normal, lo cual se confirmó durante el análisis exploratorio mediante inspección gráfica y pruebas de normalidad (véase apartado I.1). El test de Wilcoxon, al operar sobre las diferencias en rangos de cada par de medidas, ofrece una alternativa robusta frente a esta falta de normalidad.

## II.- NIVEL MEDIO

---

En este nivel intermedio del estudio se abordan técnicas más sofisticadas de construcción y evaluación de modelos predictivos, ampliando las estrategias exploradas en el Nivel Básico. Es importante señalar que este bloque, al igual que el Nivel Básico, se centra en explicar decisiones metodológicas, justificar elecciones de modelos y mostrar algunos resultados preliminares que

guiaron la evolución del proyecto. Sin embargo, el estudio definitivo con la mejor configuración y métricas finales se presenta exclusivamente en el Nivel Avanzado y en el notebook de Google Colab asociado.

## II.1.- Algoritmos de selección automática de atributos relevantes

Con el objetivo de mejorar la eficiencia de los modelos y reducir el riesgo de sobreajuste, en esta etapa de desarrollo del estudio se exploraron técnicas de selección automática de atributos, que permiten identificar las variables más relevantes para la predicción de la condición diabética. Esta estrategia completa el enfoque manual realizado en la fase previa y constituye un paso clave en la optimización del rendimiento predictivo.

Para ello se utilizó el método `SelectFromModel`, un selector automático de variables implementado en la librería Scikit-learn. Este método evalúa la importancia de cada atributo en función de un estimador base, normalmente un modelo supervisado que ha sido previamente entrenado. En este estudio, se optó por usar como estimador el mismo modelo principal con el que se evaluará el rendimiento posterior (por ejemplo, regresión logística regularizada o XGBoost), lo que garantiza que la selección de variables esté alineada con la lógica interna del clasificador.

`SelectFromModel` requiere el ajuste de varios parámetros clave. Por un lado, es necesario configurar los hiperparámetros del estimador subyacente (por ejemplo, la regularización L1 o el número de árboles en un modelo de boosting), ya que estos afectan directamente al criterio de importancia que se utiliza. Por otro, debe establecerse un umbral de corte (`threshold`) que determine qué atributos se conservan: aquellos cuya importancia supere ese valor serán seleccionados, y el resto serán descartados del conjunto de entrenamiento.

Así, el selector `SelectFromModel` se integró en la configuración 2 de la fase 1 de nuestro estudio (véase apartado III.4.a), técnica demostró ser una estrategia eficaz para reducir el número de variables manteniendo o incluso mejorando el rendimiento predictivo, siendo especialmente útil con modelos sensibles al sobreajuste, como la regresión logística o los árboles poco profundos.

No obstante, también requiere precaución: si el modelo base no está bien ajustado o si se seleccionan umbrales demasiado estrictos, existe el riesgo de eliminar variables informativas. Por ello, la selección de atributos se aplicó dentro de un proceso sistemático de validación, y sus efectos fueron evaluados empíricamente en cada combinación de configuración y clasificador. Esta integración controlada permitió explotar los beneficios de la selección automática sin comprometer la capacidad del modelo para detectar patrones relevantes en los datos.

## II.2.- Clasificadores

En esta fase intermedia del estudio se amplió la exploración de modelos predictivos, incorporando nuevos clasificadores y aplicando técnicas sistemáticas de ajuste de hiperparámetros. A los dos modelos ya utilizados en el Nivel Básico —regresión logística y Random Forest— se sumaron tres clasificadores adicionales: XGBoost, CatBoost y un Stacking Ensemble, todos ellos bien establecidos en la literatura de clasificación sobre conjuntos de datos desbalanceados. Sin embargo, por cuestiones que se comentarán más adelante, acabamos por descartar los dos últimos modelos (Catboost y el Stacking Ensemble) y optamos por quedarnos con Balanced Random Forest y LightGBM. En esta etapa de desarrollo del análisis se empleó `GridSearchCV` para la búsqueda hiperparámetros. El proceso de ajuste de modelos se desarrolló sobre cada configuración

experimental y se adaptó a las particularidades de cada algoritmo. Aunque en esta sección se exponen los procedimientos y resultados correspondientes al Nivel Medio, la implementación concreta y detallada de ciertos clasificadores, junto con técnicas de optimización más avanzadas, será tratado con mayor profundidad en el bloque correspondiente al Nivel Avanzado. A continuación, se exponen las características y criterios por los que se seleccionaron estos modelos de clasificación para este estudio.

### II.2.a.- Dummy Classifier (librería Scikit-learn)

Como paso inicial en la evaluación de modelos, se empleó un clasificador dummy (DummyClassifier), cuya función es establecer una línea base trivial que permita comparar el rendimiento de los modelos posteriores. Este tipo de clasificador no utiliza ninguna de las variables predictoras disponibles; en su lugar, se limita a aplicar reglas simples como la predicción aleatoria o constante en función de la distribución observada de clases en el conjunto de entrenamiento.

En este estudio, se utilizó la estrategia "most\_frequent", que consiste en predecir siempre la clase mayoritaria. En el conjunto de datos empleado, el 86,07% de los registros corresponden a personas sin diabetes o prediabetes, por lo que el clasificador dummy predice sistemáticamente "no diabético" para todos los casos. Como resultado, este modelo obtiene una exactitud de 0.8607, una cifra aparentemente alta, pero completamente vacía de valor clínico, ya que no detecta ni un solo caso de diabetes. Su rendimiento en recall, precisión y F1-score sobre la clase positiva es nulo, y su AUC se mantiene en el valor mínimo aceptable para un clasificador binario (0.50), lo que indica ausencia total de capacidad discriminativa.

El uso del clasificador dummy resulta esencial como modelo de referencia negativo: cualquier modelo más avanzado debe, al menos, superar este rendimiento basal. Además, su comportamiento refuerza una idea clave en este estudio: la exactitud puede ser una métrica engañosa en conjuntos de datos desbalanceados. Un modelo que simplemente predice siempre la clase más frecuente puede parecer eficaz, cuando en realidad ignora por completo a la población que más interesa detectar. Este razonamiento justifica la decisión metodológica de priorizar métricas como el AUC y el F1-score en la evaluación comparativa de clasificadores, ya que reflejan mejor el desempeño sobre la clase minoritaria (diabetes/prediabetes).

En definitiva, el clasificador dummy proporciona un punto de partida fundamental para validar el valor añadido de los modelos posteriores. Un modelo que no supera de forma clara al dummy en términos de AUC o recall, difícilmente podrá considerarse útil en un escenario clínico como el abordado en este estudio.

### II.2.b.- Logistic Regression (librería Scikit-learn)

La regresión logística es uno de los algoritmos de clasificación más conocidos y utilizados para problemas de salida binaria, como la predicción de la presencia o ausencia de diabetes. Este modelo estima la probabilidad de ocurrencia de un evento (en este caso, que una persona pertenezca a la clase positiva) en función de una combinación lineal de las variables predictoras, transformada posteriormente mediante una función logística sigmoidal, que restringe la salida al intervalo [0, 1].

Desde el punto de vista interpretativo, la regresión logística permite entender cómo variables como la edad, el índice de masa corporal (BMI), la percepción del estado de salud o el nivel de ingresos contribuyen a aumentar o reducir el riesgo de desarrollar diabetes. Esta capacidad para proporcionar probabilidades interpretables y coeficientes fácilmente analizables la convierte en una herramienta

particularmente valiosa en contextos clínicos, donde la transparencia del modelo puede ser tan importante como su rendimiento.

En el contexto de este estudio, se seleccionó la regresión logística como uno de los modelos de referencia por varias razones. En primer lugar, el conjunto de datos, compuesto por variables codificadas numéricamente y sin valores perdidos, se ajusta adecuadamente a los requisitos del modelo. En segundo lugar, se trata de un clasificador rápido de entrenar y eficiente en recursos computacionales, aunque puede verse afectada por multicolinealidad, especialmente al interpretar los coeficientes. Por ello, se evaluó previamente la redundancia entre variables antes del ajuste y posteriormente se comparó su rendimiento ante distintas técnicas de tratamiento de los datos. Además, a diferencia de otros algoritmos más complejos, permite ajustar el umbral de decisión sobre la probabilidad estimada, lo que resulta útil cuando se desea priorizar la sensibilidad sobre la precisión (por ejemplo, en estrategias de cribado poblacional). No obstante, como ocurre con todos los modelos lineales basados en coeficientes, es sensible a la escala de las variables, por lo que se aplicó un proceso de estandarización previa (transformación de media cero y desviación estándar uno). Esta transformación permite que los coeficientes aprendidos sean comparables en magnitud y mejora la convergencia del algoritmo de optimización.

Una de las limitaciones conocidas de la regresión logística es su tendencia a favorecer la clase mayoritaria en conjuntos de datos desbalanceados, como ocurre aquí, donde solo el 13,93% de los registros corresponde a personas con diabetes o prediabetes. En este escenario, un modelo entrenado sin ajustes tendería a predecir "no diabético" por defecto, optimizando la exactitud a costa de ignorar la clase minoritaria.

Para mitigar este sesgo, se implementaron diferentes técnicas de balanceo de clases durante el entrenamiento. Por un lado, se empleó el parámetro `class_weight='balanced'` para ajustar automáticamente la función de coste y dar más peso a los errores cometidos sobre la clase positiva. Por otro lado, también se exploraron enfoques basados en submuestreo o sobremuestreo del conjunto de entrenamiento, integrados en pipelines junto con el modelo. Estas estrategias permitieron mejorar considerablemente la capacidad del clasificador para detectar pacientes con diabetes, incrementando notablemente el recall y el F1-score, aunque con una precisión moderada, como es habitual en contextos con desbalance de clases..

La regresión logística, pese a su sencillez, sirvió como un punto de comparación robusto y competitivo frente a modelos más complejos. En la fase de validación cruzada, sus resultados demostraron que, con un preprocessamiento adecuado y técnicas de balanceo bien aplicadas, es posible alcanzar un rendimiento notable incluso con un modelo lineal. Su uso fue clave para entender la relación directa entre variables sociodemográficas y riesgo de diabetes, y constituye una base sólida sobre la que evaluar mejoras posteriores con algoritmos más sofisticados.

### II.2.c.- Random Forest Classifier (librería Scikit-learn)

El Random Forest es un algoritmo de clasificación basado en ensambles de árboles de decisión, ampliamente utilizado por su robustez y capacidad para modelar relaciones no lineales sin requerir transformaciones complejas de los datos. En este método, se entrena múltiples árboles sobre distintas muestras aleatorias del conjunto de entrenamiento (técnica de *bagging*), y cada árbol considera solo un subconjunto aleatorio de variables en cada nodo de decisión. Finalmente, la predicción se realiza mediante votación mayoritaria entre los árboles.

Esta estrategia introduce variabilidad controlada en los modelos base, lo que reduce la correlación entre árboles individuales y mejora la generalización. Aunque cada árbol puede sobreajustarse

ligeramente, el promedio de muchos árboles contribuye a estabilizar las predicciones, mitigando la influencia de ruido o valores atípicos. En el contexto de este estudio, el Random Forest permite detectar combinaciones complejas de variables que podrían indicar riesgo de diabetes, como la interacción entre edad, nivel de ingresos y percepción del estado de salud, que un modelo lineal difícilmente captaría.

El modelo fue especialmente útil debido al carácter tabular del conjunto de datos, compuesto por variables sociodemográficas codificadas numéricamente, muchas de ellas binarias o ordinales. El Random Forest no requiere normalización previa, tolera bien la presencia de variables redundantes y permite obtener una estimación de la importancia de cada atributo en la predicción, lo que resulta muy valioso para interpretar qué factores están más asociados al diagnóstico de diabetes o prediabetes en la muestra analizada.

Sin embargo, este modelo también presenta limitaciones. Por un lado, su costo computacional es considerable: entrenar cientos de árboles y combinar sus resultados implica tiempos de entrenamiento más largos y mayor uso de memoria en comparación con modelos más simples. Aunque este impacto fue gestionable en este estudio, en implementaciones clínicas reales con actualización continua del modelo podría ser una variable a tener en cuenta. Por otro lado, su interpretabilidad a nivel individual es limitada: si bien se pueden conocer las variables más relevantes globalmente, resulta difícil explicar por qué un paciente específico fue clasificado como positivo o negativo, debido a la naturaleza agregada del modelo.

Además, al igual que otros clasificadores no adaptados explícitamente al desbalance de clases, un Random Forest estándar tiende a favorecer la clase mayoritaria, ya que su criterio de partición en los árboles individuales (como el índice de Gini) no está diseñado para tener en cuenta el desbalance entre clases. Esto se evidenció en las primeras ejecuciones sin ajustes, donde el modelo mostró buen AUC pero una baja recuperación de la clase positiva (recall). Para abordar esta limitación, se aplicaron varias estrategias: el ajuste del parámetro `class_weight='balanced'`, el uso de submuestreo del conjunto de entrenamiento, y la integración del modelo dentro de pipelines con técnicas de rebalanceo. Estas modificaciones lograron aumentar significativamente la sensibilidad hacia los casos de diabetes, mejorando métricas clave como el F1-score sin comprometer en exceso la precisión.

En conjunto, el Random Forest se reveló como un modelo sólido en este estudio, con buen rendimiento general, capacidad para manejar relaciones complejas entre variables, y utilidad práctica tanto en predicción como en análisis exploratorio de factores asociados a la diabetes.

## II.2.d.- Balanced Random Forest Classifier (librería Imbalanced-learn)

El Balanced Random Forest (BRF), disponible en la librería imbalanced-learn, es una variante del Random Forest específicamente diseñada para conjuntos de datos con clases desbalanceadas, como el que se emplea en este estudio, donde solo el 13,93% de los registros corresponden a personas con diabetes o prediabetes.

A diferencia del Random Forest estándar, en el que cada árbol se entrena sobre una muestra bootstrap representativa del conjunto completo (y, por tanto, dominada por la clase mayoritaria), el Balanced Random Forest equilibra internamente las clases en cada árbol. Para ello, selecciona todos los casos de la clase minoritaria (diabéticos) y extrae aleatoriamente, con reemplazo, un número igual de casos de la clase mayoritaria (no diabéticos). Con este subconjunto 50/50 se entrena un

árbol. Este proceso se repite para cada árbol del bosque, de modo que cada árbol ve una fracción distinta de los negativos pero todos los positivos, garantizando así una exposición balanceada en el entrenamiento.

Esta estrategia combina las ventajas del submuestreo (reducción del sesgo hacia la mayoría) con el poder del ensamble, permitiendo entrenar múltiples modelos equilibrados que, en conjunto, mantienen una buena cobertura de la población original. El mecanismo de votación final entre árboles es el mismo que en el Random Forest tradicional, pero los árboles individuales están mejor calibrados para distinguir entre ambas clases.

En el marco de este estudio, el uso del Balanced Random Forest permitió mejorar significativamente la sensibilidad hacia la clase positiva, uno de los principales objetivos del proyecto. Al reducir la tendencia del modelo a favorecer la clase mayoritaria, el BRF fue capaz de identificar con mayor eficacia a los pacientes en riesgo de diabetes, logrando así mejores valores de recall y F1-score que otros modelos no balanceados, sin necesidad de aplicar técnicas adicionales de sobremuestreo o ajuste de pesos.

Como cabría esperar, esta mejora en la capacidad de detección de casos positivos puede ir acompañada de un ligero aumento en el número de falsos positivos, lo que reduce marginalmente la precisión. Sin embargo, este compromiso resulta clínicamente aceptable, especialmente en contextos de cribado, donde es preferible generar algunas alertas falsas a pasar por alto personas que podrían beneficiarse de una evaluación médica.

En cuanto al tiempo de ejecución, el Balanced Random Forest mostró un coste computacional alto, superior al de otros modelos más complejos como XGBoos. Esto se debe a que el muestreo interno balanceado introduce una sobrecarga adicional en cada árbol.

En definitiva, el Balanced Random Forest ofreció un rendimiento competitivo en este proyecto, con especial eficacia en la detección de pacientes con diabetes, y se consolidó como una opción altamente competitiva en este estudio para abordar el desbalance de clases sin necesidad de modificar el conjunto de entrenamiento externamente. Su implementación directa y su buena integración con pipelines de validación lo convierten en una herramienta muy adecuada para tareas de clasificación médica con datos tabulares.

## II.2.e.- Light Gradient Boosting Machine (librería LightGBM)

LightGBM es un algoritmo de boosting basado en árboles de decisión, diseñado para ofrecer un rendimiento competitivo con un consumo muy eficiente de recursos. Pertenece a la familia de los métodos de Gradient Boosted Decision Trees, como XGBoost o CatBoost, pero se distingue por una serie de optimizaciones que lo hacen más rápido, ligero y escalable. Estas características motivaron su inclusión en este estudio, especialmente dadas las exigencias computacionales derivadas del uso de validación cruzada, sobremuestreo y ajuste de hiperparámetros en un conjunto de datos amplio y desbalanceado.

LightGBM construye secuencialmente árboles donde cada nuevo modelo corrige los errores del anterior (principio fundamental del boosting), pero lo hace empleando un enfoque leaf-wise, en lugar del crecimiento nivel a nivel tradicional. En concreto, expande preferentemente aquellas hojas cuya división produce mayor ganancia de información, permitiendo reducir el error de forma más agresiva en regiones complejas del espacio de características. Esta estrategia, aunque muy eficaz, puede generar ramas desbalanceadas y profundas si no se limita el número de hojas o la profundidad

máxima (num\_leaves, max\_depth), por lo que en este estudio se controlaron cuidadosamente dichos parámetros durante la fase de ajuste.

Otra innovación clave es su uso de histogramas discretizados, que agrupan los valores numéricos en intervalos antes de buscar los puntos de corte, reduciendo significativamente el coste de cálculo. Aunque LightGBM admite variables categóricas internamente, en este estudio no fue necesario utilizar esta funcionalidad ya que todas las variables se encontraban previamente codificadas de forma numérica. Además, el modelo permite aplicar técnicas como *subsampling* de columnas (feature\_fraction) para evitar sobreajuste. Estas propiedades hacen que el algoritmo sea particularmente eficiente para conjuntos de datos tabulares con muchas variables, como el utilizado en este estudio, donde se combinan indicadores de salud, variables ordinales y características binarias.

En lo relativo al desequilibrio de clases, LightGBM dispone de mecanismos nativos para compensar el sesgo hacia la clase mayoritaria. En este proyecto se utilizaron los parámetros scale\_pos\_weight y is\_unbalance=True para incrementar el coste de los errores cometidos sobre la clase positiva (diabetes). Por ejemplo, con un 13,93% de positivos en el conjunto de datos, se estableció un scale\_pos\_weight en torno a 6.2, lo que permitió mejorar la sensibilidad y el F1-score sin necesidad de duplicar físicamente los casos minoritarios. Esta estrategia fue especialmente útil para reducir la tendencia del modelo a infrapredicir la clase diabética, un aspecto crítico en tareas de cribado médico.

Durante el desarrollo del estudio, LightGBM demostró excelente capacidad de generalización y rápida convergencia, se situó entre los modelos con mejor rendimiento global, logrando valores muy competitivos de AUC y F1-score, especialmente cuando se combinó con técnicas de selección automática de atributos (SelectFromModel) y validación cruzada estratificada. Aunque su proceso de ajuste de hiperparámetros es más complejo que el de modelos como la regresión logística, su rendimiento justifica ampliamente la inversión computacional. Además, LightGBM es compatible con herramientas de interpretabilidad como SHAP, que fueron utilizadas en fases posteriores del estudio (ver apartado III.5 del Nivel Avanzado) para explorar la contribución de las variables en las decisiones del modelo.

Entre sus limitaciones se encuentran el riesgo de sobreajuste si no se regulariza adecuadamente, su menor interpretabilidad directa y la necesidad de cierta precaución con los valores extremos, ya que la agrupación en histogramas puede atenuar o distorsionar los efectos de los outliers. No obstante, como parte del preprocesamiento del presente estudio, se gestionaron adecuadamente los valores atípicos, garantizando que el rendimiento del modelo no se viera comprometido por este aspecto.

En conjunto, LightGBM se consolidó como uno de los modelos más eficaces y eficientes del estudio, capaz de manejar con solvencia el desbalance de clases, capturar relaciones no lineales complejas y escalar bien en entornos con múltiples configuraciones de entrenamiento y validación sin sacrificar tiempo de cómputo.

## II.2.f.- eXtreme Gradient Boosting (librería XGBoost)

XGBoost es un algoritmo de boosting basado en árboles que destaca por su alto rendimiento predictivo, capacidad de regularización y eficiencia computacional. Es una implementación optimizada del clásico Gradient Boosting, que añade múltiples mejoras para aumentar su robustez frente al sobreajuste y acelerar el entrenamiento. Estas cualidades lo han convertido en una de las herramientas más utilizadas en competiciones de machine learning y lo hacen especialmente adecuado para tareas complejas como la predicción de diabetes con clases desbalanceadas.

El principio de funcionamiento es el mismo que en otros métodos de boosting: los árboles se construyen secuencialmente, y cada uno intenta corregir los errores cometidos por los anteriores. Sin embargo, XGBoost introduce regularización explícita (L1 y L2) en la función de pérdida para penalizar la complejidad de los árboles, lo que ayuda a evitar el sobreajuste, especialmente en conjuntos de datos con ruido o muchas variables correlacionadas. Además, incorpora mejoras como submuestreo de filas (subsample) y columnas (colsample\_bytree) para reducir la varianza del modelo y mejorar la generalización.

En este estudio, XGBoost fue utilizado como uno de los clasificadores avanzados dentro del bloque de Nivel Medio, con un enfoque particular en su capacidad para manejar el desbalance de clases. Para ello, se empleó el parámetro scale\_pos\_weight, que permite aumentar el coste de los errores cometidos sobre la clase minoritaria (diabetes/prediabetes). En función del ratio de clases (86,07% negativos vs. 13,93% positivos), se fijó un valor de scale\_pos\_weight ≈ 6.2, logrando así una mejora considerable en recall y F1-score sin necesidad de duplicar instancias o alterar el conjunto de entrenamiento.

Además de su buen rendimiento, XGBoost se adapta bien al pipeline de validación cruzada y ajuste de hiperparámetros implementado en este proyecto. Aunque su entrenamiento puede ser un poco más costoso que el de modelos como LightGBM, se logró mantener tiempos de ejecución muy competitivos gracias al uso de estrategias de parada anticipada (early\_stopping\_rounds). Entre los parámetros ajustados destacan la profundidad máxima de los árboles (max\_depth), el número de árboles (n\_estimators), la tasa de aprendizaje (learning\_rate) y los coeficientes de regularización (lambda, alpha), todos ellos clave para equilibrar sesgo y varianza del modelo.

Una de las ventajas adicionales de XGBoost es su capacidad para manejar directamente valores ausentes, aprendiendo de forma automática la mejor dirección de bifurcación para las muestras con datos faltantes. Aunque en este estudio el preprocesamiento de la fuente del conjunto de datos ya había eliminado o gestionado adecuadamente esos casos, esta funcionalidad refuerza su idoneidad para contextos clínicos donde los registros incompletos son frecuentes.

En cuanto a interpretabilidad, como cualquier modelo de boosting, XGBoost no permite entender de forma directa las reglas que conducen a una predicción individual. No obstante, se puede complementar con técnicas de interpretación como SHAP (SHapley Additive Explanations), que se utilizarán en el bloque de Nivel Avanzado para analizar la contribución específica de cada variable a las decisiones del modelo.

En conjunto, XGBoost demostró ser uno de los clasificadores más potentes del estudio, destacando especialmente en sensibilidad y AUC, métricas clave para la detección de casos de diabetes. Aunque su ajuste requiere cierta complejidad, los beneficios obtenidos en términos de capacidad predictiva y manejo del desbalance justifican plenamente su inclusión en el análisis comparativo.

## II.2.g.- Modelos descartados: CatBoost y Stacking Ensemble

A pesar del reconocido rendimiento de CatBoost en conjuntos de datos con variables categóricas, este modelo fue descartado en nuestro proyecto por motivos de compatibilidad, redundancia y eficiencia computacional. Dado que nuestras variables categóricas ya habían sido transformadas mediante codificación ordinal o binaria, y que todas las variables eran del tipo int64, las ventajas específicas de CatBoost —como su capacidad para manejar directamente datos categóricos sin preprocesamiento— no suponían un valor añadido relevante. Además, su tiempo de entrenamiento resultó considerablemente superior al de otros algoritmos evaluados, lo cual lo hacía poco práctico en el entorno de Google Colab, donde los recursos computacionales disponibles son limitados y las

sesiones pueden desconectarse tras un periodo de inactividad. Por tanto, se priorizó el uso de modelos más eficientes y compatibles con el flujo de trabajo ya establecido, como XGBoost y LightGBM.

En cuanto al modelo de Stacking Ensemble, su descarte se fundamentó tanto en criterios metodológicos como computacionales. Aunque esta técnica permite combinar múltiples clasificadores para mejorar el rendimiento global, su implementación requiere entrenar varios modelos base y un modelo meta, lo que multiplica el tiempo de cómputo y complica el proceso de validación cruzada, especialmente con técnicas como SMOTE aplicadas en cada partición. Esta carga computacional adicional resultó ser incompatible con los recursos gratuitos de Google Colab, dificultando la ejecución completa del pipeline sin interrupciones. Además, los modelos individuales ya optimizados mostraban métricas sólidas (AUC elevadas y estabilidad entre particiones), por lo que el esfuerzo adicional necesario para configurar un stacking no se justificaba frente a la mejora potencial, especialmente en un contexto donde la interpretabilidad y la eficiencia eran prioritarias.

### II.3.- Ajuste básico de hiperparámetros: Grid Search

En una primera aproximación, se decidió emplear la técnica clásica de búsqueda en cuadrícula (Grid Search) para el ajuste de hiperparámetros. Este método consiste en definir un conjunto discreto de valores posibles para cada parámetro del modelo y evaluar de forma exhaustiva todas las combinaciones posibles dentro de esa rejilla. Cada combinación se evalúa mediante validación cruzada, y se selecciona aquella que maximiza la métrica objetivo (en este caso, el AUC).

Por ejemplo, para ajustar una regresión logística se probó un conjunto acotado de valores del parámetro de regularización C (p. ej., [0.01, 0.1, 1, 10]) y distintos tipos de penalización (penalty='l1' o 'l2'). Para modelos como Random Forest, se exploraron configuraciones de hiperparámetros como n\_estimators, max\_depth o min\_samples\_split, combinando diferentes valores para cada uno. En esta fase inicial del estudio, los modelos eran relativamente simples y el número de combinaciones manejable, lo que permitía aplicar Grid Search sin que el coste computacional fuera prohibitivo.

Una de las razones por las que se eligió Grid Search fue su carácter exhaustivo: al recorrer sistemáticamente todas las combinaciones definidas por el usuario, ofrece garantías de no omitir ninguna posibilidad dentro de ese espacio. Además, su implementación a través de GridSearchCV de scikit-learn permite integrar fácilmente la búsqueda con técnicas de validación cruzada estratificada, como las empleadas en este proyecto.

No obstante, esta metodología mostró rápidamente sus limitaciones de escalabilidad a medida que se avanzó hacia clasificadores más complejos (como XGBoost o LightGBM), que involucran muchos más hiperparámetros y posibles valores por parámetro. El número total de combinaciones crece de forma exponencial: por ejemplo, un espacio de 5 valores por 6 hiperparámetros implica 15.625 combinaciones, lo que, con validación cruzada de 10 particiones, supondría más de 150.000 entrenamientos. Con un conjunto de datos como el empleado en este estudio (más de 250.000 registros), esta carga computacional es claramente inviable sin acceso a recursos de cómputo avanzados.

Otra desventaja es que Grid Search opera exclusivamente sobre los valores discretos definidos por el usuario. Si el valor óptimo de un hiperparámetro no está exactamente en la rejilla, no será descubierto. Por ejemplo, si learning\_rate=0.07 es el óptimo, pero solo se han probado 0.05 y 0.1, se obtendrá una solución subóptima. Además, dedica recursos a evaluar combinaciones irrelevantes

cuando alguno de los parámetros tiene escasa influencia en el resultado, lo que lo hace ineficiente frente a métodos que priorizan exploración inteligente del espacio de búsqueda.

Por estas razones, el uso de Grid Search se limitó a la fase inicial del proyecto, en modelos con pocos hiperparámetros. A medida que el estudio avanzó y se incorporaron clasificadores más complejos, Grid Search fue descartado en favor de métodos más escalables y eficientes. En el Nivel Avanzado de esta memoria se describen en detalle dos enfoques más potentes que resolvieron estas limitaciones: Randomized Search y Optuna, que permitieron explorar el espacio de hiperparámetros de forma más inteligente y adaptativa, manteniendo el rendimiento computacional dentro de márgenes razonables.

## II.4.- Ponderación de clases mediante funciones de coste

En este estudio, uno de los retos fundamentales fue abordar el fuerte desequilibrio de clases presente en el conjunto de datos, en el que solo el 13,93% de los registros corresponden a personas con diabetes o prediabetes. Frente a este tipo de desbalance, existe el riesgo de que los clasificadores aprendan a favorecer sistemáticamente la clase mayoritaria (no diabéticos), lo cual genera modelos con alta exactitud aparente pero pobre capacidad de detección de los casos realmente positivos.

Además del uso de técnicas explícitas de balanceo como oversampling o undersampling, se implementó una estrategia complementaria: el ajuste de funciones de coste mediante la ponderación de clases. Esta técnica consiste en modificar la forma en que el algoritmo penaliza los errores cometidos en cada clase, asignando un mayor coste a los errores sobre la clase minoritaria. De esta forma, el modelo presta más atención a los casos positivos, sin necesidad de alterar físicamente el conjunto de entrenamiento.

Para ello, se emplearon los siguientes hiperparámetros, según el modelo:

- `class_weight='balanced'` (Scikit-learn): disponible en modelos como Logistic Regression y Random Forest, este parámetro ajusta automáticamente los pesos de las clases de forma inversamente proporcional a su frecuencia en los datos. Esto significa que, durante el entrenamiento, se penaliza más un error cometido sobre un paciente con diabetes que sobre uno saludable. Esta opción fue sencilla de aplicar y produjo mejoras claras en sensibilidad y F1-score, especialmente en modelos lineales como la regresión logística, que de otro modo tenderían a ignorar la clase minoritaria.
- Balanced Random Forest (librería imbalanced-learn): este clasificador incorpora internamente el balanceo de clases mediante submuestreo en cada árbol. Por tanto, no requiere ni acepta el uso explícito de ponderaciones. En su lugar, cada árbol se entrena sobre una muestra bootstrap equilibrada, formada por todos los casos positivos y una cantidad igual de negativos seleccionados aleatoriamente. Esta estrategia permite mejorar la detección de la clase minoritaria sin necesidad de ajustar una función de pérdida ponderada.
- `scale_pos_weight` en XGBoost: este hiperparámetro permite especificar de forma manual el peso de la clase positiva en la función de pérdida. Se calculó como la razón entre los casos negativos y positivos (`N_negativos / N_positivos`), dando como resultado un valor aproximado de 6.2 para este dataset. Este ajuste permitió a XGBoost prestar más atención a los casos de diabetes durante el entrenamiento, incrementando su sensibilidad sin necesidad de sobremuestrear los datos.
- `scale_pos_weight` en LightGBM: inicialmente se probó el uso de este parámetro con el mismo objetivo que en XGBoost, pero se observaron resultados inestables. En algunas combinaciones, el uso de `scale_pos_weight` en conjunción con `SelectFromModel` provocó

problemas de estabilidad, ya que la estimación de la importancia de variables puede verse afectada por la ponderación, generando una selección excesivamente restrictiva y dejando matrices con columnas vacías. Debido a este comportamiento, se optó finalmente por utilizar la alternativa `is_unbalance=True`, que permite a LightGBM ajustar internamente los pesos sin requerir un valor manual.

En la mayoría de los casos, el ajuste de pesos a través de estas funciones de coste se reveló como una solución eficaz y eficiente para mejorar el rendimiento de los modelos, especialmente en lo que respecta a la detección de la clase minoritaria. Esta estrategia permitió reducir la dependencia de técnicas más costosas computacionalmente como el sobremuestreo, y se integró con éxito en el flujo de trabajo de validación cruzada y ajuste de hiperparámetros. En general, los clasificadores con ponderación de clases mostraron mejoras consistentes en recall y AUC, ofreciendo una alternativa robusta y eficiente frente al oversampling, aunque no siempre alcanzaron el mejor resultado absoluto en todas las métricas.

## II.5.- Comparación de modelos mediante pruebas estadísticas

Para asegurar que las diferencias observadas en el rendimiento de los distintos modelos no fueran producto del azar o de la variabilidad inherente al proceso de validación cruzada, se aplicaron pruebas estadísticas formales de contraste de hipótesis. El objetivo de este análisis fue determinar si las diferencias entre los modelos eran estadísticamente significativas, y por tanto, si podían respaldar la elección de un modelo sobre otro con base empírica rigurosa.

Para ello, se desarrolló una función auxiliar denominada `perform_stat_tests()`, que opera sobre el diccionario `cv_scores`. Este diccionario recopila las métricas de evaluación (principalmente AUC) obtenidas por cada modelo durante la validación cruzada estratificada, almacenando los resultados de cada partición (`fold`) para todos los modelos probados.

La función ejecuta diferentes pruebas estadísticas según el número y tipo de modelos a comparar:

- Prueba de Friedman: se utilizó cuando se compararon más de dos modelos evaluados sobre los mismos subconjuntos de datos (es decir, cuando las observaciones están relacionadas). Esta prueba no paramétrica es una alternativa al ANOVA de medidas repetidas cuando no puede asumirse normalidad en las distribuciones, como es el caso en el presente estudio. Se aplicó, por ejemplo, para comparar simultáneamente los AUC de cinco clasificadores (Regresión Logística, Random Forest, Balanced RF, LightGBM y XGBoost) sobre las 10 particiones de validación cruzada. La hipótesis nula es que todos los modelos tienen el mismo rendimiento. Un valor  $p$  bajo indica que al menos uno de los modelos difiere significativamente.
- Prueba de Wilcoxon para muestras pareadas: cuando se trataba de comparar dos modelos concretos entre sí, se empleó esta prueba no paramétrica para datos emparejados, ideal para comparar métricas obtenidas por cada modelo sobre los mismos folds de validación. Para mantener el control sobre el error de tipo I al realizar múltiples comparaciones, se aplicó una corrección de Bonferroni, ajustando el umbral de significación en función del número de pares comparados. Esta corrección permitió identificar qué pares de modelos presentaban diferencias estadísticamente significativas en su rendimiento.
- Prueba de Kruskal-Wallis: utilizada en casos donde se necesitaba comparar más de dos grupos independientes (por ejemplo, diferentes configuraciones o estrategias de

preprocesamiento aplicadas a modelos distintos, sin compartir los mismos folds). Esta prueba permite evaluar si existen diferencias significativas entre grupos sin asumir normalidad ni igualdad de varianzas. Aunque menos habitual en este estudio (ya que se empleó validación cruzada estratificada común en la mayoría de los casos), también estuvo disponible como recurso para análisis comparativos más amplios.

- Prueba U de Mann-Whitney: utilizada cuando se compararon dos modelos evaluados sobre conjuntos de validación independientes, es decir, sin relación directa entre las observaciones. Esta prueba no paramétrica determina si las puntuaciones de una muestra tienden a ser mayores que las de otra. En este estudio se empleó en casos puntuales donde, por la estructura del experimento (por ejemplo, uso de diferentes técnicas de preprocesamiento o distintos conjuntos de variables), no era posible emparejar directamente los folds de validación entre modelos.

Estas pruebas estadísticas se complementaron con representaciones visuales (diagramas de caja, curvas ROC superpuestas, y gráficos de barras de medias con errores estándar) que ayudaron a interpretar los resultados de forma intuitiva y a reforzar la comprensión de las diferencias entre modelos. Gracias a este análisis, fue posible no solo identificar qué modelos ofrecían mejor desempeño promedio, sino también establecer cuáles de esas diferencias eran estadísticamente significativas, dando soporte cuantitativo a la elección final del modelo óptimo.

En conclusión, el uso sistemático de pruebas estadísticas permitió que la selección de los mejores clasificadores en cada fase del estudio se basara no solo en observaciones empíricas, sino en evidencia estadística sólida, garantizando la validez de las decisiones tomadas a lo largo del proyecto.

## II.6.- Métricas de evaluación de la clasificación binaria

En tareas de clasificación binaria con fuerte desequilibrio entre clases, como ocurre en este estudio, donde los casos de diabetes o prediabetes representan solo el 13,93% del total, no todas las métricas de evaluación ofrecen una imagen fiel del rendimiento del modelo.

Por ello, a lo largo de este estudio se han utilizado métricas específicas que permiten evaluar con mayor precisión la capacidad del modelo para identificar correctamente los casos positivos, sin ignorar los negativos. La elección de estas métricas también responde a la necesidad de comparar modelos de forma objetiva, especialmente en presencia de técnicas de balanceo de clases, ajuste de pesos o métodos de selección de atributos.

A continuación, se describen las cinco métricas principales utilizadas en este proyecto para evaluar el rendimiento de los clasificadores, con especial atención a su interpretación en el contexto del desequilibrio de clases.

### II.6.a.- Exactitud (Accuracy)

La *exactitud* es la proporción de aciertos del modelo sobre el total de ejemplos evaluados. Formalmente,  $\text{Accuracy} = (\text{VP} + \text{VN}) / (\text{VP} + \text{VN} + \text{FP} + \text{FN})$ , donde VP son *verdaderos positivos*, VN *verdaderos negativos*, FP *falsos positivos* y FN *falsos negativos*. Es decir, mide qué fracción de las clasificaciones (positivas o negativas) resultaron correctas. Un modelo perfecto tendría una exactitud de 1.0 (100% aciertos).

La exactitud ofrece una visión general del rendimiento del modelo y es especialmente útil cuando las clases están equilibradas y los costes de los distintos errores son similares. En esos casos, puede

actuar como una métrica resumen válida, fácil de interpretar y comunicar. De hecho, muchas herramientas de aprendizaje automático la utilizan como métrica por defecto.

Sin embargo, su utilidad se reduce considerablemente en contextos con clases desbalanceadas, como ocurre en este estudio. En tales situaciones, un modelo que simplemente predice siempre la clase mayoritaria ("no diabético") puede obtener una alta exactitud ( $\approx 86\%$ ) sin detectar ni un solo caso positivo. Este comportamiento, aunque aparentemente satisfactorio desde el punto de vista de la métrica, resulta clínicamente inaceptable.

Además, la exactitud no diferencia entre tipos de error. En contextos médicos, un falso negativo (no identificar a una persona con diabetes) suele tener consecuencias más graves que un falso positivo (diagnosticar erróneamente a una persona sana). La exactitud, al tratar ambos errores por igual, no refleja esta asimetría en el impacto clínico, lo que limita su valor como criterio principal de evaluación.

Durante el desarrollo de esta etapa del estudio, se observó cómo algunos clasificadores, como la regresión logística sin ponderación de clases, lograban valores altos de exactitud simplemente aprendiendo a predecir mayoritariamente la clase negativa. Sin embargo, su rendimiento en términos de recall (sensibilidad) y AUC era bajo, ya que ignoraban sistemáticamente los casos positivos. En contraste, modelos como Random Forest con submuestreo o XGBoost ajustado con `scale_pos_weight` presentaban una leve disminución en exactitud, pero mejoraban significativamente en métricas centradas en la detección de pacientes diabéticos.

Un ejemplo claro se observó con Balanced Random Forest, cuyo mecanismo interno fuerza el equilibrio entre clases. Este modelo sacrificaba algo de exactitud global al cometer más falsos positivos, pero conseguía identificar una mayor proporción de casos positivos, lo que se reflejaba en mejores valores de recall y AUC.

Por estas razones, a lo largo de la práctica, la exactitud se ha reportado como métrica complementaria, pero no se ha utilizado como criterio principal para seleccionar los modelos óptimos. En un problema centrado en la detección de enfermedades, métricas como recall, F1-score o AUC ofrecen una imagen mucho más fiable del rendimiento real del sistema predictivo.

## II.6.b – Precisión (Precision)

La *precisión* (precision) mide la proporción de predicciones positivas que realmente son casos positivos. En términos formales,  $\text{Precisión} = \text{VP} / (\text{VP} + \text{FP})$ . Equivale al valor predictivo positivo: indica cuántos de los sujetos que el modelo marcó como "diabéticos" efectivamente tienen diabetes. Un modelo con precisión perfecta (1.0) no cometería falsos positivos, es decir, no etiquetaría incorrectamente a personas sanas como enfermas.

Esta métrica responde a la pregunta: "De todos los pacientes que el modelo ha marcado como diabéticos, ¿qué proporción lo son realmente?". En el contexto clínico, esto equivale al valor predictivo positivo: si un modelo tiene una precisión del 85%, significa que 85 de cada 100 pacientes etiquetados como positivos por el modelo realmente presentan la condición.

La precisión es especialmente relevante cuando los falsos positivos generan consecuencias costosas, invasivas o emocionalmente perjudiciales. Por ejemplo, si la predicción de diabetes conlleva repetir análisis de laboratorio, realizar una prueba de tolerancia a la glucosa o iniciar seguimiento clínico, una precisión elevada evita someter a personas sanas a procesos innecesarios.

En esta etapa del estudio, se observaron diferencias claras entre clasificadores. Por ejemplo, la regresión logística mostró un comportamiento más conservador, generando menos positivos con mayor precisión, pero a costa de pasar por alto muchos casos (bajo recall). Modelos como Balanced Random Forest o XGBoost con ajustes de clase tendieron a ser más agresivos en la detección de positivos, lo que aumentó el recall pero redujo la precisión, ya que también se incrementaron los falsos positivos.

Aunque es una métrica útil, la precisión puede resultar engañosa en problemas con clases muy desequilibradas, como este. En estas condiciones, un modelo puede lograr alta precisión simplemente siendo muy conservador, es decir, prediciendo positivos solo cuando está extremadamente seguro, lo que reduce la cobertura. Esto suele ir en detrimento de la sensibilidad. Por ejemplo, un modelo que predice “diabetes” solo en casos muy evidentes podría alcanzar un 95% de precisión, pero ignorar al 70% de los pacientes realmente diabéticos.

Además, cuando hay pocos positivos en el conjunto de datos, pequeñas variaciones en el número de falsos positivos pueden alterar significativamente la precisión, haciéndola estadísticamente inestable. Por ello, se recomienda interpretarla siempre junto con otras métricas complementarias, especialmente el recall y la F1-score.

Durante este proyecto, la precisión fue clave para evaluar la fiabilidad de los positivos predichos por cada modelo. Por ejemplo, un modelo con alta sensibilidad pero baja precisión podría saturar el sistema sanitario con falsas alarmas, sometiendo a personas sanas a evaluaciones innecesarias. Por el contrario, un modelo con alta precisión pero baja sensibilidad podría no detectar a la mayoría de pacientes en riesgo, lo cual también es problemático.

En definitiva, la precisión fue considerada una métrica fundamental cuando se buscaba minimizar los falsos positivos, por ejemplo en escenarios con presupuesto limitado para seguimientos o pruebas complementarias. No obstante, se empleó en combinación con otras métricas, ya que la optimización exclusiva de la precisión puede llevar a ignorar demasiados casos reales de diabetes.

### II.6.c.- Sensibilidad (Recall)

La *sensibilidad* (también llamada *recall* o tasa de verdaderos positivos) mide la capacidad del modelo de identificar los casos positivos entre todos los positivos reales. Formalmente, es  $\text{Recall} = \text{VP} / (\text{VP} + \text{FN})$ . Representa la fracción de pacientes con la condición (diabetes) que el modelo logra detectar correctamente. Un modelo con sensibilidad 1.0 (100%) identificaría todos los positivos ( $\text{VP} = \text{positivos reales}$ ,  $\text{FN} = 0$ ). Esta métrica también se conoce como *probabilidad de detección*.

La sensibilidad adquiere especial relevancia en contextos clínicos y de cribado, donde no detectar un caso positivo puede tener consecuencias serias. En este tipo de aplicaciones, se considera preferible generar falsas alarmas que pasar por alto pacientes que podrían beneficiarse de un tratamiento temprano. En este estudio, dado que la clase positiva (diabetes/prediabetes) representa solo el 13,93% de los registros, la sensibilidad es una métrica prioritaria, ya que se focaliza directamente en el desempeño del modelo sobre la clase minoritaria.

Sin embargo, optimizar exclusivamente la sensibilidad puede comprometer otras métricas, como la precisión. Un modelo que predice siempre “diabetes” obtendrá una sensibilidad perfecta, pero generará muchos falsos positivos y tendrá una precisión muy baja. Este es el motivo por el que la sensibilidad no se ha considerado de forma aislada, sino siempre junto a métricas complementarias como la precisión o el F1-score.

Durante esta etapa del estudio, se observó que ciertos modelos eran capaces de mejorar significativamente la sensibilidad mediante ajustes específicos. La regresión logística, cuando se entrenaba sin ponderación de clases, tenía tendencia a ser conservadora y mostraba una sensibilidad reducida. En cambio, al aplicar técnicas de balanceo, como el submuestreo o el uso del parámetro `class_weight='balanced'`, la capacidad de detectar positivos mejoraba notablemente. Modelos como LightGBM y XGBoost, al incorporar mecanismos como `scale_pos_weight`, también lograron incrementos sustanciales en sensibilidad. En particular, el modelo Balanced Random Forest, al entrenar cada árbol con una muestra equilibrada de positivos y negativos, mostró una sensibilidad significativamente más alta, aunque con un aumento esperado de falsos positivos.

Por ejemplo, un modelo que alcanzaba una sensibilidad del 90%, lo que significa que de cada 100 personas con diabetes, el modelo lograba identificar correctamente a 90. La pregunta clave es si resulta aceptable dejar sin detectar al 10% restante. En el contexto de la diabetes, donde la intervención temprana puede evitar complicaciones a largo plazo, este nivel de sensibilidad puede considerarse razonable, especialmente si los falsos positivos solo llevan pruebas adicionales de confirmación.

En definitiva, la sensibilidad ha sido una métrica central para evaluar la capacidad de los modelos en detectar la condición de interés. En un problema como el presente, en el que lo más costoso es no identificar a un paciente enfermo, se ha priorizado esta métrica por encima de otras más generales como la exactitud. La sensibilidad ha influido tanto en la comparación entre clasificadores como en la selección final del modelo más adecuado para su uso en contextos sanitarios.

## II.6.d.- Puntuación F1 (F1-score)

El *F1-Score* es la media armónica de la precisión y la sensibilidad. Se calcula como  $F1 = 2 * (\text{Precisión} * \text{Sensibilidad}) / (\text{Precisión} + \text{Sensibilidad})$ . En términos de VP, FP, FN, la fórmula se puede escribir como  $F1 = 2 \cdot VP / (2 \cdot VP + FP + FN)$ .

El F1 toma un valor alto solo cuando ambas métricas son elevadas. Si una de ellas es baja, el F1 desciende rápidamente. Por ejemplo, un modelo con precisión y sensibilidad ambas iguales a 0.80 tendría un F1 de 0.80, mientras que otro con precisión 0.95 pero sensibilidad 0.50 tendría un F1 reducido a aproximadamente 0.66, penalizando ese desequilibrio.

Esta métrica es especialmente útil en contextos con clases desbalanceadas, como el presente estudio sobre predicción de diabetes, donde la clase positiva representa solo el 13,93% del total. A diferencia de la exactitud, el F1-Score no se ve inflado por el gran número de verdaderos negativos, y por tanto ofrece una visión más realista del rendimiento del modelo sobre la clase que más nos interesa detectar.

El F1 es especialmente valioso para comparar modelos con distintos compromisos entre precisión y sensibilidad. Por ejemplo, si un modelo mejora mucho la sensibilidad pero pierde algo de precisión (o viceversa), el F1 permite capturar ese equilibrio general en un único valor. Este equilibrio es especialmente relevante en problemas de salud, donde es necesario detectar al mayor número posible de pacientes con enfermedad, pero también evitar falsas alarmas en exceso.

Sin embargo, el F1 también tiene limitaciones. Su fórmula asume que la precisión y la sensibilidad son igual de importantes, lo cual no siempre es cierto en aplicaciones clínicas. Si, por ejemplo, se prioriza explícitamente la sensibilidad (como ocurre a menudo en cribados de enfermedades),

podrían preferirse métricas ajustadas o medidas separadas. Además, el F1 no ofrece una interpretación directa para un profesional sanitario: no especifica cuántos casos positivos se están detectando ni cuántas alertas son acertadas, por lo que suele acompañarse de los valores individuales de precisión y recall.

En el estudio se utilizaron varios ejemplos prácticos para interpretar esta métrica. Por ejemplo, se observó que un modelo con precisión 0.70 y sensibilidad 0.70 alcanzaba un F1 de 0.70, mientras que otro con precisión 0.85 pero sensibilidad 0.55 solo lograba un F1 de aproximadamente 0.67, evidenciando su desequilibrio. A nivel práctico, un F1 alto (por ejemplo, 0.78) indica que el modelo logra identificar correctamente una alta proporción de los pacientes con diabetes, sin generar un número excesivo de falsos positivos.

Durante la comparación de modelos en esta etapa del estudio, el F1-Score fue una métrica clave para seleccionar configuraciones con buen rendimiento global. Por ejemplo, un Balanced Random Forest —aunque con menor precisión— logró un F1 superior al de un Random Forest estándar, al recuperar más casos positivos. En otro ejemplo, una regresión logística con precisión 0.60 y sensibilidad 0.40 ofrecía un F1 de 0.48, mientras que un Balanced RF con precisión 0.50 y sensibilidad 0.70 alcanzaba un F1 cercano a 0.58, demostrando que el equilibrio entre ambas dimensiones compensa individualmente valores extremos.

Modelos como LightGBM y XGBoost no optimizan directamente el F1, pero es posible evaluar esta métrica sobre el conjunto de validación para seleccionar el mejor modelo. También se pueden ajustar umbrales de decisión para maximizar el F1 o adaptarlo a las prioridades clínicas concretas.

Cabe destacar que en el estudio definitivo, el F1-score no se empleó como métrica principal para la comparación sistemática entre modelos durante las fases intermedias (validación cruzada y selección de clasificadores). Su uso se reservó para la etapa final del pipeline, una vez seleccionado el modelo definitivo, con el objetivo de ajustar el umbral de decisión que maximiza el compromiso entre precisión y sensibilidad. Esta distinción metodológica permitió mantener la coherencia del proceso comparativo (basado en AUC) y aplicar F1-score como criterio clínicamente relevante en el ajuste fino del clasificador final.

## II.6.e – Área bajo la curva ROC (AUC-ROC)

El AUC-ROC (Área bajo la curva Receiver Operating Characteristic) es una métrica que resume la capacidad del modelo para discriminar entre clases a lo largo de todos los posibles umbrales de decisión. La curva ROC representa gráficamente la relación entre la Tasa de Verdaderos Positivos (sensibilidad) y la Tasa de Falsos Positivos (1 - especificidad), a medida que se varía el umbral de clasificación. El AUC es el valor del área bajo esa curva, y oscila entre 0.5 (equivalente a clasificar al azar) y 1.0 (discriminación perfecta).

Desde un punto de vista intuitivo, el AUC puede interpretarse como la probabilidad de que el modelo asigne una puntuación de riesgo más alta a un paciente con diabetes que a uno sin ella, si ambos se seleccionan aleatoriamente. Así, un modelo con AUC = 0.85 sugiere que, en el 85% de las parejas positivas/negativas, el caso positivo será correctamente priorizado por el modelo.

Una de las principales ventajas del AUC-ROC es que no depende de un umbral específico para clasificar. Esto permite evaluar la calidad del modelo de forma global, sin necesidad de comprometerse con un punto de corte arbitrario. Esta propiedad lo convierte en una métrica particularmente útil para comparar modelos entre sí, ya que refleja su capacidad intrínseca de separar las clases, independientemente de cómo se ajuste luego el umbral en producción.

El AUC-ROC es también relativamente robusto frente al desbalance de clases, como ocurre en este estudio. A diferencia de la exactitud, que puede verse fuertemente influida por la abundancia de negativos, el AUC considera proporciones relativas (tasas), por lo que un modelo con AUC elevado sugiere que consigue asignar consistentemente puntuaciones más altas a los casos positivos, aun cuando estos sean minoría.

No obstante, el AUC también tiene limitaciones. Aunque ofrece una evaluación general del modelo, no garantiza un buen rendimiento en un umbral concreto. Dos modelos con el mismo AUC pueden comportarse de manera muy diferente si se analiza su precisión o sensibilidad en un punto específico de decisión. Además, cuando la clase positiva es extremadamente rara, el AUC puede parecer alto incluso cuando el modelo tiene una baja precisión práctica. Por eso, en contextos con prevalencias muy bajas, puede ser recomendable complementar el análisis con curvas Precision-Recall. En este caso, con una prevalencia positiva cercana al 14%, la curva ROC sigue ofreciendo una imagen fiel del rendimiento.

En esta etapa del estudio, el AUC-ROC fue la métrica principal para comparar el rendimiento de los distintos clasificadores. Su independencia del umbral permitió valorar el poder discriminativo de modelos como regresión logística, Random Forest, LightGBM o XGBoost, incluso cuando se aplicaban técnicas de sobremuestreo o ajuste de pesos de clase. Por ejemplo, un modelo con  $AUC = 0.90$  podía interpretarse como una excelente herramienta de clasificación, muy por encima del nivel de azar, mientras que otro con  $AUC = 0.75$  ofrecía una discriminación aceptable pero mejorable.

En muchos casos, se observaron mejoras significativas de AUC al pasar de modelos básicos a técnicas más sofisticadas. Así, si la regresión logística obtenía un AUC de 0.78, y un modelo de boosting como XGBoost alcanzaba 0.88, esto indicaba una mejora sustancial en la capacidad para distinguir entre pacientes con y sin diabetes. Estas diferencias, además, son consistentes con lo que suele considerarse en la literatura médica:  $AUC > 0.8$  como buen rendimiento diagnóstico, entre 0.7–0.8 como moderado, y por debajo de 0.6 como pobre.

A pesar de su utilidad como métrica comparativa, el AUC no debe usarse en solitario para tomar decisiones clínicas. Un modelo con alto AUC puede no ofrecer el mejor balance entre sensibilidad y precisión en el punto de operación seleccionado. Por eso, una vez identificado el modelo con mayor AUC, se exploraron diferentes umbrales de decisión para encontrar el que mejor se ajustara a las necesidades clínicas (por ejemplo, priorizar sensibilidad en contextos de cribado).

En definitiva, el AUC-ROC ha sido una herramienta central de evaluación en este estudio, ya que proporciona una medida global, objetiva y robusta del rendimiento de los modelos ante un conjunto de datos desbalanceado. Su uso ha permitido comparar con claridad el valor predictivo de diferentes técnicas y establecer cuáles ofrecen mejor capacidad de discriminación global para ser consideradas en aplicaciones reales de detección de diabetes.

### III.- NIVEL AVANZADO

---

Este último bloque de la memoria presenta el estudio definitivo de los modelos predictivos desarrollados, integrando todas las decisiones metodológicas tomadas a lo largo del proyecto y consolidando los análisis con mayor rigor técnico y profundidad. A diferencia de los Niveles Básico y Medio —donde se exponen opciones preliminares, justificaciones conceptuales y resultados exploratorios—, el Nivel Avanzado recoge exclusivamente las configuraciones óptimas seleccionadas tras un proceso sistemático de validación cruzada, búsqueda de hiperparámetros y ajuste del umbral

de decisión. Asimismo, se incorporan herramientas avanzadas de interpretabilidad como SHAP y se aplican pruebas estadísticas no paramétricas para fundamentar empíricamente las comparaciones entre modelos. Este bloque también documenta las estrategias de modularización del código y los procedimientos computacionales utilizados para garantizar la eficiencia y reproducibilidad del análisis. En definitiva, este nivel representa la culminación del proyecto, donde se definen y validan los modelos finales que podrían trasladarse a un entorno clínico real.

### III.1.- Técnicas de reducción de la dimensionalidad: UMAP

En la última fase de desarrollo del proyecto, se exploró la posibilidad de reducir la dimensionalidad del conjunto de datos como estrategia de preprocesamiento previa a la clasificación. Aunque el número total de variables predictoras (21) no es excesivo, muchas de ellas son categóricas binarias o presentan correlaciones entre sí, lo que sugiere una posible redundancia. Esto motivó la evaluación de técnicas que permitieran comprimir el espacio de representación sin comprometer la capacidad predictiva del modelo.

Entre las alternativas disponibles, se descartaron enfoques clásicos como el Análisis de Componentes Principales (PCA) y el Análisis Discriminante Lineal (LDA), por sus limitaciones en este contexto. PCA, aunque útil para visualizar patrones de varianza en los datos, es una técnica lineal no supervisada que no considera la clase objetivo, y puede distorsionar relaciones importantes cuando se trata de tareas de clasificación. LDA, por otro lado, sí tiene en cuenta la pertenencia a clases y busca proyectar los datos maximizando su separabilidad. Sin embargo, parte de supuestos muy restrictivos (como la normalidad multivariada y la igualdad de varianzas y covarianzas entre grupos) que no se verifican en este conjunto de datos, según quedó patente en el análisis exploratorio.

Por esta razón se optó por utilizar UMAP (Uniform Manifold Approximation and Projection), una técnica de reducción de dimensionalidad no lineal que ha ganado popularidad en los últimos años. A diferencia de PCA o LDA, UMAP no presupone relaciones lineales entre las variables. En su lugar, se basa en principios de topología algebraica y teoría de grafos para construir una representación de menor dimensión que preserve tanto la estructura local como la global del conjunto de datos original. En la práctica, UMAP construye un grafo de similitud entre las muestras en el espacio de alta dimensión y luego proyecta ese grafo en un nuevo espacio, típicamente de dos dimensiones, optimizando la conservación de las relaciones de vecindad.

La elección de UMAP estuvo motivada por su capacidad para capturar relaciones complejas entre las variables predictoras y su habilidad para representar patrones no lineales que podrían ser relevantes para la clasificación de pacientes según su riesgo de diabetes. Se implementó como parte de la configuración 3 en la Fase 1 del estudio, transformando el conjunto completo de variables en una representación bidimensional que sirviera como entrada a los clasificadores.

No obstante, los resultados no fueron los esperados. Los modelos entrenados sobre las proyecciones generadas por UMAP obtuvieron puntuaciones sustancialmente inferiores en comparación con los modelos entrenados sobre el conjunto completo de variables originales o tras aplicar técnicas de selección automática como SelectFromModel. Métricas como el AUC, el F1-score y el recall de la clase positiva disminuyeron de forma notable, lo que evidenció una pérdida significativa de capacidad predictiva. Este comportamiento puede deberse a varias razones. Por un lado, al reducir el espacio a solo dos dimensiones, es probable que se haya perdido información relevante contenida en variables clave. Por otro, UMAP está diseñado principalmente para tareas de visualización exploratoria, y no garantiza preservar la separabilidad necesaria para fines de clasificación si no se emplea junto con técnicas supervisadas. Además, su comportamiento puede variar dependiendo de

los parámetros empleados (`n_neighbors`, `min_dist`), así como del valor de la semilla aleatoria, ya que UMAP no es determinista si no se controla explícitamente este aspecto.

En definitiva, aunque UMAP representa una técnica poderosa para representar visualmente estructuras de alta dimensión y explorar agrupamientos latentes, no resultó adecuada como paso previo a la clasificación en este caso concreto. La experiencia refuerza una lección metodológica importante: las transformaciones de los datos deben evaluarse no solo por su sofisticación teórica, sino por su efecto empírico en el rendimiento del modelo final. En este estudio, las configuraciones basadas en selección de variables relevantes conservaron una mayor capacidad discriminativa, lo que confirma que, en problemas de predicción clínica como este, la reducción agresiva de dimensiones puede resultar contraproducente si no se justifica con una ganancia clara de desempeño o interpretabilidad.

### III.2.- Modularización y funciones auxiliares

Conforme avanzaba el desarrollo del proyecto y se consolidaban las técnicas que formarían parte del análisis definitivo, fue necesario estructurar y modularizar el código para garantizar una ejecución más eficiente y un mantenimiento más claro. Para ello, se implementó un conjunto de funciones auxiliares reutilizables, organizadas en la sección “Funciones Auxiliares” del cuaderno de Google Colab, dentro del bloque 3 “Implementación de modelos predictivos”. Estas funciones automatizan tareas comunes a todos los clasificadores, permitiendo una arquitectura de código más robusta, limpia y escalable.

#### III.2.a.- Selección de núcleos para parallelización

Para mejorar los tiempos de ejecución, se especificó que las tareas computacionales se ejecutasesen en paralelo utilizando todos los núcleos disponibles del entorno. Esto se logró definiendo `n_jobs = -1`, instrucción que indica a scikit-learn y a otros frameworks compatibles que utilicen todos los hilos del sistema. Esta decisión fue especialmente útil al entrenar modelos con validación cruzada o al aplicar algoritmos de búsqueda de hiperparámetros que ejecutan múltiples combinaciones en paralelo.

#### III.2.b.- Estrategia de validación cruzada

Se definió una estrategia uniforme de validación cruzada mediante `StratifiedKFold` con 10 particiones. Este método asegura que la distribución de la variable objetivo (diabético/no diabético) se conserve en cada subconjunto, lo que es fundamental al trabajar con clases desbalanceadas. Además, se utilizó una semilla fija (`random_state`) para garantizar la reproducibilidad de los resultados, de modo que cada partición pudiera replicarse en futuras ejecuciones.

#### III.2.c.- Cálculo de métricas de evaluación

La función `evaluate_and_return_metrics()` centraliza el proceso de evaluación de los modelos entrenados. Recibe como entrada el mejor modelo tras el ajuste de hiperparámetros (`best_estimator`), los datos de prueba (`X_test` o `X_test_filtered`), las etiquetas reales (`y_test` o `y_test_filtered`), el tiempo de ejecución (`execution_time`), los hiperparámetros a optimizar (`best_params`), los resultados de la validación cruzada (`cv_scores`) y, opcionalmente, un umbral de decisión (`threshold`) para convertir probabilidades en etiquetas. Si no se indica, se asume por defecto un umbral de 0.5. La función devuelve un diccionario con métricas clave como AUC, exactitud, precisión, recall, matriz de confusión, probabilidades predichas, tiempo de ejecución y estadísticas de validación cruzada (media y desviación estándar del AUC), además de imprimir un resumen en pantalla. Esta salida estandarizada permite comparar fácilmente los modelos evaluados.

### III.2.d.- Visualización comparativa de resultados

Para facilitar la comparación entre modelos, la función `plot_results()` toma un diccionario con los resultados de `evaluate_and_return_metrics()` para todos los modelos evaluados, y genera una tabla resumen de métricas. Además, representa gráficamente las matrices de confusión de cada modelo, lo que permite identificar de forma visual los aciertos y errores de cada clasificador en la predicción de casos positivos y negativos.

### III.2.e.- Análisis estadístico de los modelos

La función `perform_stat_tests()` se encarga de realizar pruebas no paramétricas de contraste de hipótesis para comparar estadísticamente el rendimiento de los distintos modelos, utilizando los valores de AUC obtenidos durante la validación cruzada estratificada. Esta función opera sobre un diccionario `cv_scores` que almacena, para cada modelo, los AUC obtenidos en los mismos 10 folds, lo que permite aplicar pruebas para datos emparejados.

En primer lugar, se aplica la prueba de Friedman, adecuada para comparar más de dos modelos evaluados sobre subconjuntos relacionados. Esta prueba no paramétrica permite detectar si existen diferencias globales de rendimiento entre los clasificadores. Si el resultado es significativo ( $p < 0.05$ ), se procede a realizar comparaciones post-hoc.

Para las comparaciones par a par se utiliza Wilcoxon signed-rank test para muestras emparejadas, también en su versión unilateral (`alternative='greater'`), con el fin de evaluar si un modelo supera significativamente a otro. En este contexto, se aplicó la corrección de Bonferroni para ajustar los p-valores obtenidos y controlar el riesgo de error de tipo I.

Cuando se realizan múltiples comparaciones, como ocurre al comparar por pares los 6 modelos finales (lo que implica 15 pruebas independientes), aumenta la probabilidad de obtener diferencias significativas por azar. Este fenómeno, conocido como error familiar de tipo I, puede llevar a conclusiones incorrectas si no se corrige. La corrección de Bonferroni consiste en dividir el nivel de significancia estándar ( $\alpha = 0.05$ ) por el número de comparaciones realizadas, lo que endurece el criterio para considerar una diferencia como estadísticamente significativa. Esta estrategia aporta mayor rigor metodológico, asegurando que las diferencias detectadas entre modelos sean fiables y no producto del azar.

Además, como complemento exploratorio, se aplicaron también dos pruebas diseñadas para muestras independientes: la prueba de Kruskal-Wallis (para comparar más de dos grupos) y el test U de Mann-Whitney (o Wilcoxon rank-sum, para comparaciones binarias) también con corrección de Bonferroni. Aunque en este estudio los datos son emparejados, estas pruebas se incluyeron como referencia cruzada para verificar la robustez de los hallazgos, siendo sus resultados interpretados con cautela.

Finalmente, los resultados de todas las pruebas se imprimen por pantalla. Gracias a este enfoque, fue posible fundamentar estadísticamente las decisiones sobre qué modelos ofrecían un rendimiento significativamente superior, añadiendo una capa de validación empírica sólida a la comparación de clasificadores realizada en el proyecto.

### III.2.f.- Optimización del umbral para maximizar el F1-Score

Con el objetivo de obtener el mejor compromiso entre precisión y recall, se definió la función `maximize_f1_score()`. Esta función toma el modelo final entrenado y explora distintos valores de umbral de decisión para convertir probabilidades en etiquetas binarias, con el objetivo de maximizar

el F1-Score. Una vez identificado el umbral óptimo, este se reutiliza en `evaluate_and_return_metrics()` para recalcular las métricas. Además, se genera un gráfico que muestra cómo varía el F1-Score en función del umbral, lo que facilita la interpretación visual de este proceso de ajuste.

### III.2.g.- Interpretabilidad con SHAP

Para aportar explicaciones interpretables a las predicciones de los modelos, se implementó la función `plot_shap()`, que calcula e ilustra los valores SHAP. Esta función recibe el modelo entrenado, los conjuntos de entrenamiento y prueba, así como el tipo de modelo ('linear' o 'tree') para elegir el método de explicación apropiado. Devuelve tres visualizaciones estándar: un gráfico de barras con la importancia media de cada variable, un gráfico de dispersión SHAP summary (con puntos coloreados por valor de la variable) y un gráfico de cascada que permite interpretar predicciones individuales. Esta función es clave para dar sentido clínico a los resultados y justificar las decisiones del modelo ante expertos no técnicos.

## III.3.- Buscadores de hiperparámetros avanzados

Uno de los principales cuellos de botella detectados en las primeras fases del proyecto fue el elevado coste computacional asociado al uso de Grid Search para el ajuste de hiperparámetros. Si bien esta técnica garantiza una búsqueda exhaustiva dentro del espacio definido, su escalabilidad se ve gravemente limitada cuando el número de hiperparámetros o de valores por hiperparámetro crece. En modelos complejos como los algoritmos de boosting o en pipelines con múltiples transformaciones, el número de combinaciones posibles puede alcanzar cifras prohibitivas.

Para superar esta limitación, se optó por integrar en el análisis definitivo dos técnicas de optimización más eficientes, que permiten explorar el espacio de búsqueda de forma mucho más rápida y efectiva sin comprometer significativamente la calidad de los resultados. Estas técnicas, implementadas y documentadas en el cuaderno de Google Colab, son Randomized Search y Optuna, cada una con sus ventajas y particularidades. Ambos métodos se describen en los subapartados siguientes.

### III.3.a.- Randomized Search (librería Scikit-Learn)

La búsqueda aleatoria de hiperparámetros, o Randomized Search, es una estrategia que permite explorar de forma eficiente grandes espacios de hiperparámetros sin incurrir en los costes computacionales de una búsqueda exhaustiva. A diferencia de Grid Search, que prueba todas las combinaciones posibles de una cuadrícula predefinida, Randomized Search selecciona aleatoriamente un número limitado de combinaciones dentro de los rangos o distribuciones especificadas para cada hiperparámetro. De esta manera, permite recorrer regiones más amplias del espacio de búsqueda con una fracción del tiempo y esfuerzo computacional.

En el contexto de este estudio, esta técnica resultó especialmente útil al aumentar la complejidad de los modelos y el número de hiperparámetros implicados, como sucedía con Random Forest, Balanced Random Forest, LightGBM o XGBoost, donde ajustar únicamente dos o tres hiperparámetros ya generaba un número elevado de combinaciones. Con `RandomizedSearchCV` (de la librería Scikit-learn), fue posible limitar el número de iteraciones a un valor manejable (por ejemplo, 25 o 50), conservando al mismo tiempo la posibilidad de encontrar combinaciones de hiperparámetros competitivas.

Cada iteración consiste en una configuración única de hiperparámetros, seleccionada al azar. Esta se evalúa con validación cruzada estratificada de 10 particiones (10-fold CV), empleando como métrica principal el AUC. En este estudio, se incluyeron en la búsqueda tanto hiperparámetros clásicos (como

`n_estimators, max_depth, min_samples_split, learning_rate)` como parámetros sensibles al desbalance de clases, tales como `class_weight`, `scale_pos_weight` o `is_unbalance`, dependiendo del clasificador utilizado. De este modo, se evaluaban no solo las capacidades estructurales del modelo, sino también su sensibilidad frente al desequilibrio de clases.

Uno de los beneficios más significativos de Randomized Search fue su flexibilidad en cuanto al presupuesto computacional. A diferencia de Grid Search, donde el número de combinaciones crece exponencialmente con el número de hiperparámetros, aquí se podía controlar explícitamente cuántas configuraciones evaluar (p. ej., `n_iter=25`), equilibrando así el tiempo de entrenamiento con la calidad esperada de los resultados. Esta capacidad fue crítica en este proyecto, donde muchos modelos se entrenaban sobre un dataset amplio y desequilibrado, con validación cruzada de 10 pliegues, además de no contar con recursos de ejecución ilimitados dado el entorno en el que trabajamos (Google Colab).

Además, Randomized Search ha demostrado ser más efectivo que Grid Search en escenarios con múltiples hiperparámetros donde solo unos pocos son realmente determinantes. Tal como muestran estudios como el de Bergstra y Bengio (2012), las búsquedas aleatorias tienden a asignar más combinaciones a los parámetros importantes por simple probabilidad, lo que incrementa las chances de encontrar buenas configuraciones en menos tiempo. En este estudio, se observó que incluso con menos iteraciones que Grid Search, se lograban AUC similares o superiores en menos tiempo.

Por ejemplo, en la búsqueda de hiperparámetros para Balanced Random Forest, Randomized Search permitió evaluar distintos valores de `n_estimators`, `max_depth`, `min_samples_leaf`, así como diferentes métodos de muestreo interno. En el caso de XGBoost, se probaron tasas de aprendizaje (`learning_rate`), número de árboles (`n_estimators`), fracciones de submuestreo (`subsample`, `colsample_bytree`), y regularizaciones (`reg_alpha`, `reg_lambda`), incluyendo el ajuste de `scale_pos_weight` para controlar el impacto del desbalance de clases. Gracias a la aleatoriedad, muchas combinaciones no convencionales resultaron ser más eficaces que las “típicas” predefinidas en una cuadrícula.

La implementación técnica fue sencilla: la clase `RandomizedSearchCV` de Scikit-learn permitió configurar fácilmente las distribuciones de hiperparámetros a explorar, integrarse con pipelines personalizados (por ejemplo, los que contenían transformaciones de imputación o escalado), y evaluar los resultados con validación cruzada y métricas específicas (en este caso, AUC).

En resumen, Randomized Search supuso un punto de inflexión en el desarrollo de este estudio. Fue clave para continuar optimizando modelos una vez Grid Search se volvió inabordable, permitiendo avanzar con configuraciones razonables de hiperparámetros en tiempos viables, sin sacrificar la calidad del modelo. Su uso ha sido decisivo en los clasificadores más exigentes como Balanced Random Forest, LightGBM y XGBoost, permitiendo obtener modelos altamente competitivos frente a la línea base y muy superiores al clasificador dummy, con un coste computacional razonable.

### III.3.b.- Optuna (librería Optuna)

Optuna es un framework moderno y potente de optimización automática de hiperparámetros, especialmente diseñado para encontrar configuraciones óptimas de modelos complejos de manera más eficiente que los métodos clásicos como Grid Search o Randomized Search. En este estudio, se recurrió a Optuna en las etapas más avanzadas de ajuste fino, particularmente para optimizar modelos de boosting como XGBoost y LightGBM, donde el número de hiperparámetros relevantes y sus posibles interacciones hacen que otros métodos se vuelvan prohibitivamente lentos o ineficientes.

A diferencia de Grid Search o Randomized Search, Optuna no selecciona combinaciones de parámetros de forma ciega o predefinida, sino que emplea un enfoque adaptativo y secuencial: en cada iteración (denominada *trial*), el sistema utiliza la información obtenida en *trials* anteriores para decidir de manera inteligente qué combinación de hiperparámetros evaluar a continuación. El motor que guía esta búsqueda es un modelo probabilístico basado en optimización bayesiana, concretamente el algoritmo TPE (*Tree-structured Parzen Estimator*), que estima la probabilidad de que una nueva combinación de hiperparámetros mejore el rendimiento observado hasta el momento.

Este proceso es particularmente valioso en escenarios como el de este proyecto, donde cada evaluación del modelo (con validación cruzada de 10 particiones y un dataset de más de 250.000 registros) supone un coste computacional significativo. Mientras que Grid Search o Randomized Search reparten su presupuesto de evaluaciones de forma uniforme, Optuna prioriza aquellas regiones del espacio que parecen más prometedoras, permitiendo descartar rápidamente configuraciones poco eficaces y concentrar los esfuerzos en zonas de alto potencial. Esta inteligencia adaptativa se tradujo, en la práctica, en una reducción notable del número de iteraciones necesarias para alcanzar configuraciones de alto rendimiento.

Además, Optuna incorpora una funcionalidad muy relevante para contextos con recursos limitados: el early stopping automático (denominado *pruning*). Esta característica permite interrumpir ensayos poco prometedores antes de completarlos, ahorrando tiempo en configuraciones que, tras pocas rondas de entrenamiento, ya muestran rendimientos por debajo de lo esperado. Por ejemplo, durante la optimización de un modelo LightGBM con 1000 árboles, Optuna puede decidir cancelar un ensayo tras 150 árboles si detecta que el AUC en validación está muy por debajo del resto de ensayos. En este estudio, esta capacidad fue crítica para evitar cuellos de botella durante la búsqueda de hiperparámetros óptimos.

Otra ventaja relevante de Optuna es su flexibilidad y modularidad. A través de su sistema de *samplers*, permite incorporar restricciones, priorizar exploraciones específicas o definir hiperparámetros condicionales (p. ej., si se selecciona el booster gblinear, no tiene sentido ajustar parámetros relacionados con árboles). Esto permitió definir espacios de búsqueda personalizados para XGBoost y LightGBM en función de las necesidades concretas del estudio. Además, su integración con frameworks como Scikit-learn, xgboost y lightgbm facilitó la implementación sin necesidad de reescribir los pipelines ya construidos.

Durante el desarrollo de este proyecto, Optuna fue particularmente valioso en las etapas finales del ajuste del modelo XGBoost, donde el número de hiperparámetros implicados (learning rate, número de árboles, profundidad máxima, fracciones de submuestreo, regularizaciones L1 y L2, y el peso de la clase positiva scale\_pos\_weight) hacía inviable el uso de Grid Search. Gracias al enfoque bayesiano de Optuna, se logró explorar de forma eficiente este espacio de búsqueda de alta dimensión, obteniendo mejoras sustanciales en el AUC y en el equilibrio entre sensibilidad y precisión, sin necesidad de aumentar desproporcionadamente el número de pruebas.

Una desventaja de Optuna es su menor transparencia para el usuario novato. Al tratarse de un proceso guiado por modelos probabilísticos internos, las decisiones sobre qué parámetros probar no son tan intuitivas como en un Grid Search. No obstante, esta desventaja fue mitigada gracias a las herramientas de visualización integradas que Optuna proporciona, como los gráficos de evolución de la métrica objetivo, importancia de hiperparámetros o análisis de interacciones entre ellos, que se utilizaron en fases de interpretación avanzada del proyecto.

En definitiva, Optuna se consolidó como el método más eficiente y efectivo de búsqueda de hiperparámetros en este estudio, especialmente para los modelos más exigentes como XGBoost. Su capacidad de aprendizaje adaptativo, su integración fluida con los modelos empleados y su ahorro significativo en tiempo de cómputo permitieron alcanzar resultados óptimos con un coste computacional moderado, asegurando además la reproducibilidad y el control del proceso mediante el uso de semillas aleatorias y almacenamiento estructurado de los ensayos.

### III.4.- Estructura del ajuste de los modelos de clasificación

En este punto de desarrollo del estudio, la meta fue llevar a cabo un ajuste lo más robusto y exhaustivo posible para cada modelo seleccionado, intentando cubrir el máximo número de combinaciones de técnicas, así como evitando realizar cualquier asunción en la medida de lo posible. Sin embargo, se encontró que la acumulación de posibilidades técnicas ya era demasiado grande como para probarlas todas exhaustivamente, como se muestra a continuación:

- Técnicas de selección de atributos:
  1. Todos (all)
  2. Selección manual (man\_sel)
  3. Selección automática (aut\_sel)
  4. Reducción de dimensionalidad (red\_dim)
- Técnicas de equilibrado de clases:
  1. Oversampling (ov)
  2. Undersampling (un)
  3. Class\_weight (wght)
- Técnicas de búsqueda automática de hiperparámetros:
  1. RandomizedSearchCV (RandCV)
  2. Optuna (Opt)

En lugar de evaluar exhaustivamente las 24 combinaciones que surgen al cruzar estas tres dimensiones para cada uno de los modelos, se optó por dividir el proceso en dos fases diferenciadas.

#### III.4.a.- Fase 1: Tuning de los datos

En esta fase se evalúan exclusivamente las combinaciones de métodos que afectan directamente a la forma de los datos. Se emplean las cuatro técnicas de transformación de variables (usar todas, selección manual, selección automática o reducción de dimensionalidad). Para medir su rendimiento justamente, en todas las configuraciones se emplean RandomizedSearchCV y Undersampling debido a que es la combinación que menor tiempo de cómputo registra. Una vez completada esta fase, la técnica de trasformación de variables que mayor AUC haya presentado se seleccionará para implementarse en la fase 2.

#### III.4.b.- Fase 2: Tuning del modelo

Una vez que se ha determinado la configuración óptima en cuanto a los datos, se mantiene esa configuración y se prueban las combinaciones que afectan directamente el desempeño del modelo, es decir, se cruzan los dos métodos de búsqueda de hiperparámetros (RandomizedSearchCV y Optuna) con las tres técnicas de balanceo de clases (class\_weight, oversampling con SMOTE y undersampling). La combinación que dé mejores resultados en cuanto a AUC será seleccionado como el modelo final (ya que será el modelo que presentará el mejor “tuning de datos” y el mejor “tuning del modelo”).

### III.4.c.- *tuning\_data\_model()*

Para la ejecución de las fases 1 y 2 se definió la función *tuning\_data\_model()*. Esta función toma como parámetros:

- Una cadena que indica el tipo de buscador automático de hiperparámetros que se desea emplear (*searcher\_type*: 'randcv' u 'optuna').
- El pipeline que contiene el modelo junto otros pasos adicionales (*ImbPipeline*).
- Un diccionario que define el espacio de hiperparámetros (*param\_distributions*).
- Los datos de entrenamiento (*X\_train* e *y\_train* o *X\_train\_filtered* e *y\_train\_train\_filtered*).
- Los datos de prueba (*X\_test* e *y\_test* o *X\_test\_filtered* e *y\_test\_filtered*).
- Un objeto de validación cruzada (*KFold*).
- Un número de iteraciones para los buscadores de hiperparámetros (*n\_iter*).
- El número de núcleos a emplear en la búsqueda (*n\_jobs*).

La función *tuning\_data\_model()* llama al buscador de hiperparámetros solicitado, con todas las métricas ajustadas, entrena el modelo resultante, extrae el array de scores de la validación cruzada (*cv\_scores*), calcula el tiempo de ejecución (*execution\_time*) y llama a la función *evaluate\_and\_return\_metrics()* pasándole los parámetros necesarios. Como resultado, la función devuelve el diccionario que crea la función *evaluate\_and\_return\_metrics()*.

### III.4.d.- Final model

Una vez completadas las dos fases de ajuste del modelo de clasificación, se construye el modelo final utilizando el mismo modelo que aquel que mostró el mejor rendimiento en la fase 2 (que no deja de ser la mejor técnica de selección de atributos resultado de la fase 1 y la mejor técnica de equilibrado de clases resultado de la fase 2). Sin embargo, no se volverá a implementar la búsqueda de hiperparámetros puesto que esta ya ha realizado en la fase 2, sino que se indicarán en el clasificador de forma manual.

Lo que hace que este modelo sea el final y no una copia de aquel implementado en la fase 2 es la maximización del F1 score, llamando a la función *maximize\_f1\_score()* y pasándole por parámetro todo lo que necesite. Como resultado, devuelve un threshold óptimo y un gráfico de la variación del F1 score en función del mismo. Así, se llama una última vez a *evaluate\_and\_return\_metrics()* con el umbral óptimo y este devuelve las métricas definitivas del modelo final del clasificador.

También se llama a *plot\_results()* para que imprima el cuadro resumen y la matriz de confusión, así como a *plot\_shap()* para que dibuje los correspondientes gráficos explicativos.

## III.5.- SHAP

SHAP (SHapley Additive exPlanations) es una técnica de interpretabilidad basada en la teoría de juegos que permite descomponer las predicciones de un modelo de machine learning en contribuciones atribuibles a cada variable de entrada. Su fundamento se basa en los valores de Shapley, que cuantifican de manera justa y coherente cuánto aporta cada característica a la predicción final, analizando múltiples combinaciones posibles de variables. Esta metodología proporciona una herramienta potente para entender no solo qué decisión ha tomado el modelo, sino también por qué la ha tomado.

En este proyecto, SHAP se ha utilizado para explicar las predicciones de todos los modelos finales de los clasificadores empleados. Para cada modelo, se ha generado un explicador específico en función de su naturaleza (modelos basados en árboles o modelos lineales), calculando los valores SHAP a

partir de una muestra representativa de los datos (máximo 5000 observaciones para asegurar eficiencia computacional).

La función *plot\_shap()*, diseñada específicamente para este propósito, ofrece tres visualizaciones clave que permiten una comprensión profunda de los resultados del modelo:

- Summary bar plot. Este gráfico presenta la importancia media absoluta de cada variable en las predicciones del modelo. Se calcula promediando el valor absoluto de los valores SHAP de cada característica a lo largo de todas las observaciones. El resultado es un ranking de variables ordenadas por su influencia promedio, lo que permite identificar de forma clara qué factores son más relevantes a la hora de predecir si un paciente es sano o prediabético/diabético. Es especialmente útil para evaluar la importancia global de cada variable sin entrar en detalles sobre cómo afectan (positiva o negativamente) a cada predicción.
- Summary dot plot. Este gráfico es una extensión del anterior y proporciona una visión más granular. Representa cada valor SHAP como un punto en el eje horizontal, agrupado por característica en el eje vertical. Además, utiliza un gradiente de color para reflejar el valor real que toma la característica correspondiente en cada observación (por ejemplo, un nivel alto de BMI o un grupo de edad avanzado). Esto permite observar no solo cuán influyente es una variable, sino también en qué dirección influye y cómo se relaciona con sus valores. Por ejemplo, se puede ver que valores altos de una característica tienden a aumentar la probabilidad de que el modelo prediga "diabetes", mientras que valores bajos la reducen, aunque esta tendencia no siempre se cumple.
- Waterfall plot. Este gráfico se centra en una única predicción individual y muestra paso a paso cómo cada variable contribuye a mover la predicción desde la base esperada (valor promedio del modelo) hasta el resultado final. La visualización en forma de cascada descompone visualmente el impacto de cada característica: las variables que empujan la predicción hacia "diabético" aparecen en rojo, mientras que las que la empujan hacia "sano" lo hacen en azul. Esta explicación detallada resulta clave para validar casos clínicos concretos y entender por qué el modelo ha clasificado a un paciente determinado de esa manera, facilitando una interpretación a nivel individual y clínico.

Gracias a estas visualizaciones, es posible identificar los indicadores de salud más determinantes en la clasificación de pacientes como sanos o prediabéticos/diabéticos. Así, SHAP no solo aporta confianza y transparencia al uso de modelos complejos, sino que también permite una interpretación clínica más significativa de los factores subyacentes que guían la predicción del riesgo de diabetes.

Por último, cabe destacar la elección de la librería SHAP por sobre otras debido a que cuenta con implementaciones eficientes específicas (TreeExplainer) para los modelos de árbol empleados (RandomForest, LightGBM y XGBoost). Además, a diferencia de LIME, que asume independencia local, SHAP puede capturar interacciones locales y globales entre variables.

## III.6.- Resultados obtenidos

### III.6.a.- Clasificador Dummy

El Dummy Classifier es un modelo de referencia muy simple que no utiliza ninguna lógica de aprendizaje real. Su objetivo principal es establecer una línea base mínima de rendimiento que cualquier modelo más avanzado debería superar fácilmente. En este caso, se ha utilizado la

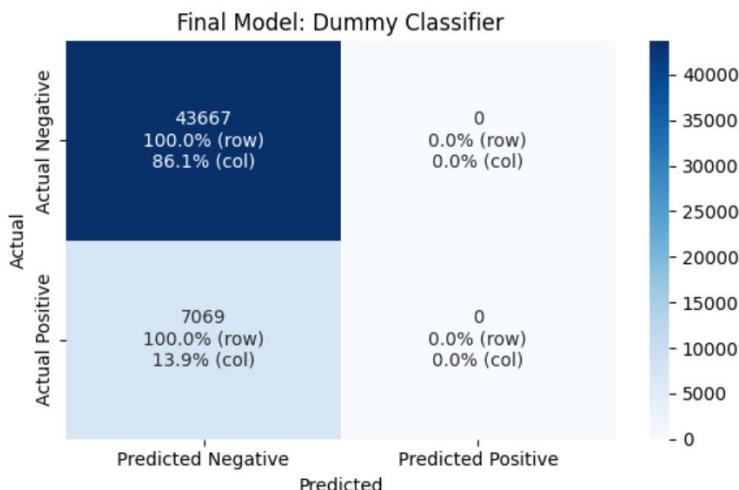
estrategia de predicción por clase más frecuente, es decir, el clasificador siempre predice la clase mayoritaria ("no diabético").

Los resultados reflejan precisamente este comportamiento: la accuracy alcanza un valor aparentemente alto (0.86), pero esta métrica es engañosa al no tener en cuenta el desbalance de clases. De hecho, tanto el recall como la precision sobre la clase positiva (diabetes/prediabetes) son nulos (0.00), ya que el modelo no identifica ningún caso positivo correctamente. Como consecuencia, el AUC se sitúa en 0.5, lo que equivale al azar, y no aporta valor predictivo real.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Final Model: Dummy Classifier	0.5	0.860671	0.0	0.0	0.5	0.0	0.00157

La matriz de confusión ilustra claramente esta situación: el clasificador acierta con todos los negativos (43.667 casos) y falla en todos los positivos (7.069 casos), que son sistemáticamente clasificados como negativos.



Este modelo base es útil como punto de partida: cualquier modelo que no supere significativamente sus resultados no se consideraría mejor que adivinar al azar. Por tanto, su principal función es servir como referencia mínima a superar por los modelos más elaborados que se desarrollarán en fases posteriores.

### III.6.b.- Logistic Regression

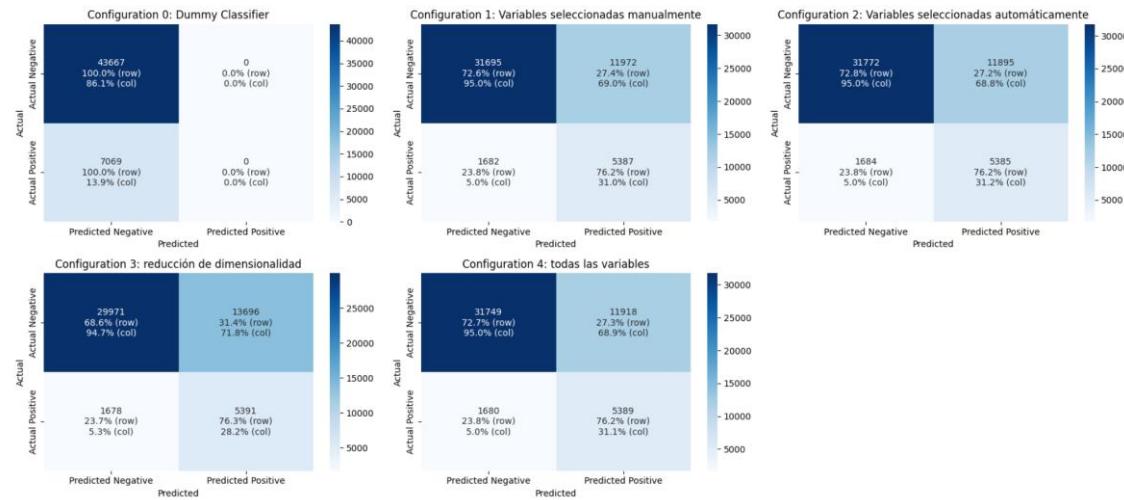
#### Fase 1

Durante la Fase 1 del ajuste de la regresión logística se exploraron diversas configuraciones de selección de variables con el objetivo de evaluar su impacto en el rendimiento predictivo del modelo.

Los mejores resultados en términos de área bajo la curva (AUC) se obtuvieron con la Configuración 2 (selección automática) y la Configuración 4 (todas las variables), ambas alcanzando un AUC de aproximadamente 0.820. Estas configuraciones no solo superaron significativamente al Dummy, sino también a la Configuración 1 (selección manual), que, aunque competitiva, mostró un leve descenso en las métricas clave. En contraste, la Configuración 3 (reducción de dimensionalidad) ofreció un rendimiento inferior, con un AUC de 0.746, indicando una posible pérdida de información relevante durante la compresión de atributos.

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Configuration 0: Dummy Classifier	0.50000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000559
1	Configuration 1: Variables seleccionadas manualmente	0.818696	0.730881	0.762060	0.310329	0.822842	0.002908	1.378730
2	Configuration 2: Variables seleccionadas automáticamente	0.820028	0.732360	0.761777	0.311632	0.824139	0.002885	2.449265
3	Configuration 3: reducción de dimensionalidad	0.746138	0.696980	0.762626	0.282444	0.761319	0.002911	143.545359
4	Configuration 4: todas las variables	0.820105	0.731985	0.762343	0.311377	0.824169	0.002991	1.447919

Las matrices de confusión permiten analizar con mayor profundidad el rendimiento por clase. Las configuraciones 1, 2 y 4 muestran una capacidad robusta para identificar correctamente los casos positivos, con un recall cercano al 76% y un porcentaje de predicciones positivas correctas (precisión) en torno al 31%, manteniendo una estabilidad notable en términos de proporción entre falsos positivos y negativos. En cambio, la Configuración 3 presenta un mayor número de falsos positivos, lo cual se refleja en una precisión reducida (28.4%), y una leve disminución en la exactitud global (69.9%).



Desde un punto de vista estadístico, los resultados obtenidos fueron analizados mediante pruebas de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman reveló diferencias significativas entre configuraciones ( $p < 0.0001$ ), y el test de Wilcoxon indicó diferencias relevantes entre la reducción de dimensionalidad y el resto de configuraciones ( $p\text{-ajustado} < 0.01$ ), especialmente con respecto a la selección automática y el uso de todas las variables. Estas diferencias también fueron confirmadas por el test de Mann-Whitney U, consolidando la decisión de no utilizar técnicas de reducción de dimensionalidad en este caso, dado su impacto negativo en el rendimiento del modelo.

Resultados del Test de Friedman:  
 Estadístico Friedman p-value  
 38.08 1.078775e-07

Resultados del Test de Wilcoxon:

	Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Configuration 0: Dummy Classifier	Configuration 1: man_sel	0.0	1.000000	1.000000	
Configuration 0: Dummy Classifier	Configuration 2: aut_sel	0.0	1.000000	1.000000	
Configuration 0: Dummy Classifier	Configuration 3: red_dim	0.0	1.000000	1.000000	
Configuration 0: Dummy Classifier	Configuration 4: all	0.0	1.000000	1.000000	
Configuration 1: man_sel	Configuration 2: aut_sel	0.0	1.000000	1.000000	
Configuration 1: man_sel	Configuration 3: red_dim	55.0	0.000977	0.009766	
Configuration 1: man_sel	Configuration 4: all	0.0	1.000000	1.000000	
Configuration 2: aut_sel	Configuration 3: red_dim	55.0	0.000977	0.009766	
Configuration 2: aut_sel	Configuration 4: all	26.0	0.577148	1.000000	
Configuration 3: red_dim	Configuration 4: all	0.0	1.000000	1.000000	

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis p-value  
 38.771191 7.767233e-08

Resultados del Test de Mann-Whitney U:

	Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Configuration 0: Dummy Classifier	Configuration 1: man_sel	0.0	0.999977	1.000000	
Configuration 0: Dummy Classifier	Configuration 2: aut_sel	0.0	0.999977	1.000000	
Configuration 0: Dummy Classifier	Configuration 3: red_dim	0.0	0.999977	1.000000	
Configuration 0: Dummy Classifier	Configuration 4: all	0.0	0.999977	1.000000	
Configuration 1: man_sel	Configuration 2: aut_sel	33.0	0.907062	1.000000	
Configuration 1: man_sel	Configuration 3: red_dim	100.0	0.000091	0.000913	
Configuration 1: man_sel	Configuration 4: all	33.0	0.907062	1.000000	
Configuration 2: aut_sel	Configuration 3: red_dim	100.0	0.000091	0.000913	
Configuration 2: aut_sel	Configuration 4: all	49.0	0.545139	1.000000	
Configuration 3: red_dim	Configuration 4: all	0.0	0.999933	1.000000	

Por todo ello, se seleccionó como configuración óptima para la siguiente fase de ajuste la Configuración 4, que emplea todas las variables originales, al ofrecer un equilibrio ideal entre rendimiento predictivo (AUC alto y score mean elevado), estabilidad (baja desviación estándar) y eficiencia computacional (tiempo de ejecución razonable de 1.45 minutos).

## Fase 2

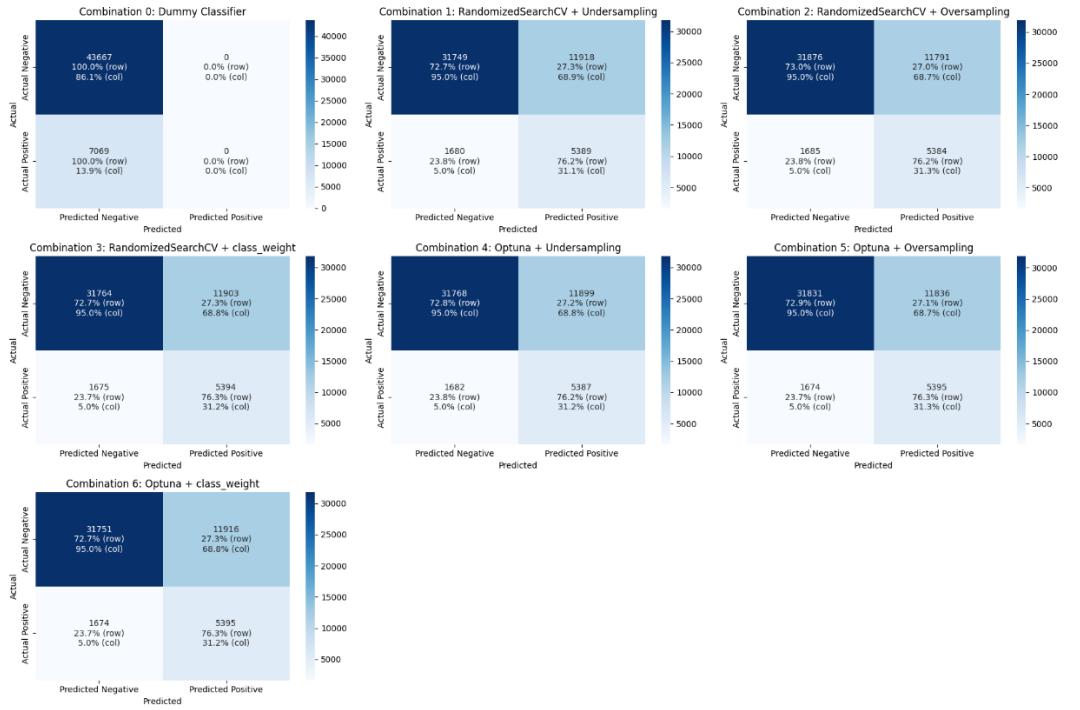
Durante la Fase 2 se exploraron distintas estrategias de ajuste fino del modelo, combinando técnicas de optimización de hiperparámetros (RandomizedSearchCV y Optuna) con métodos de balanceo de clases (undersampling, oversampling y ajuste de pesos mediante `class_weight='balanced'`). El objetivo fue evaluar si estas combinaciones mejoraban el rendimiento del modelo en comparación con la configuración base seleccionada en la fase anterior.

Los resultados mostraron una estabilidad sorprendente entre combinaciones, con valores de AUC que oscilaron entre 0.819 y 0.820, lo que indica que todas las variantes lograron un desempeño muy similar. Tanto RandomizedSearchCV como Optuna produjeron modelos robustos, y ninguna estrategia de balanceo se impuso claramente sobre las demás. En términos de ejecución, Optuna combinada con undersampling (Combinación 4) resultó ser la opción más eficiente, con un tiempo de entrenamiento inferior a 2 minutos.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Combination 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000559
1	Combination 1: RandomizedSearchCV + Undersampling	0.820105	0.731985	0.762343	0.311377	0.824169	0.002991	2.319637
2	Combination 2: RandomizedSearchCV + Oversampling	0.819491	0.734390	0.761635	0.313479	0.823526	0.003032	16.293276
3	Combination 3: RandomizedSearchCV + class_weight	0.820267	0.732379	0.763050	0.311846	0.824262	0.002896	5.589127
4	Combination 4: Optuna + Undersampling	0.820146	0.732320	0.762060	0.311639	0.824224	0.002937	1.510905
5	Combination 5: Optuna + Oversampling	0.819412	0.733720	0.763191	0.313098	0.823367	0.003060	16.397142
6	Combination 6: Optuna + class_weight	0.820208	0.732143	0.763191	0.311652	0.824204	0.002936	5.652865

Las matrices de confusión reflejan esta consistencia: en todas las combinaciones útiles (1 a 6), el modelo logró identificar correctamente aproximadamente el 76.2% de los casos positivos (recall), con una precisión en torno al 31%. Esto implica que, aunque la proporción de verdaderos positivos es elevada, sigue existiendo un volumen importante de falsos positivos, característico de problemas con clases desbalanceadas.



Desde el punto de vista estadístico, se realizaron múltiples comparaciones mediante los tests de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman confirmó la existencia de diferencias significativas entre las combinaciones ( $p < 0.0001$ ), pero los tests por pares revelaron que dichas diferencias eran en su mayoría marginales o no significativas. Algunas excepciones se observaron al comparar oversampling con undersampling (p.ej. Combination 1 vs Combination 2) y también frente a ciertas combinaciones con class\_weight, como se refleja en los valores p del test de Wilcoxon ajustados por corrección.

Resultados del Test de Friedman:  
 Estadístico Friedman p-value  
 49.628571 5.579999e-09

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 2: RandCV + Ov	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 5: Opt + Ov	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	1.000000	1.000000
Combination 1: RandCV + Un	Combination 2: RandCV + Ov	55.0	0.000977	0.020508
Combination 1: RandCV + Un	Combination 3: RandCV + wght	13.0	0.934570	1.000000
Combination 1: RandCV + Un	Combination 4: Opt + Un	20.0	0.784180	1.000000
Combination 1: RandCV + Un	Combination 5: Opt + Ov	55.0	0.000977	0.020508
Combination 1: RandCV + Un	Combination 6: Opt + wght	21.0	0.753906	1.000000
Combination 2: RandCV + Ov	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 2: RandCV + Ov	Combination 4: Opt + Un	0.0	1.000000	1.000000
Combination 2: RandCV + Ov	Combination 5: Opt + Ov	54.0	0.001953	0.041016
Combination 2: RandCV + Ov	Combination 6: Opt + wght	0.0	1.000000	1.000000
Combination 3: RandCV + wght	Combination 4: Opt + Un	35.0	0.246094	1.000000
Combination 3: RandCV + wght	Combination 5: Opt + Ov	55.0	0.000977	0.020508
Combination 3: RandCV + wght	Combination 6: Opt + wght	46.0	0.032227	0.676758
Combination 4: Opt + Un	Combination 5: Opt + Ov	55.0	0.000977	0.020508
Combination 4: Opt + Un	Combination 6: Opt + wght	31.0	0.384766	1.000000
Combination 5: Opt + Ov	Combination 6: Opt + wght	0.0	1.000000	1.000000

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis p-value  
 26.917143 0.00015

Resultados del Test de Mann-Whitney U:

Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 2: RandCV + Ov	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 5: Opt + Ov	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	0.999977	1.0
Combination 1: RandCV + Un	Combination 2: RandCV + Ov	61.0	0.213678	1.0
Combination 1: RandCV + Un	Combination 3: RandCV + wght	48.0	0.574947	1.0
Combination 1: RandCV + Un	Combination 4: Opt + Un	49.0	0.545139	1.0
Combination 1: RandCV + Un	Combination 5: Opt + Ov	62.0	0.192337	1.0
Combination 1: RandCV + Un	Combination 6: Opt + wght	49.0	0.545139	1.0
Combination 2: RandCV + Ov	Combination 3: RandCV + wght	40.0	0.786322	1.0
Combination 2: RandCV + Ov	Combination 4: Opt + Un	39.0	0.807663	1.0
Combination 2: RandCV + Ov	Combination 5: Opt + Ov	56.0	0.338792	1.0
Combination 2: RandCV + Ov	Combination 6: Opt + wght	40.0	0.786322	1.0
Combination 3: RandCV + wght	Combination 4: Opt + Un	51.0	0.484925	1.0
Combination 3: RandCV + wght	Combination 5: Opt + Ov	61.0	0.213678	1.0
Combination 3: RandCV + wght	Combination 6: Opt + wght	53.0	0.425053	1.0
Combination 4: Opt + Un	Combination 5: Opt + Ov	63.0	0.172352	1.0
Combination 4: Opt + Un	Combination 6: Opt + wght	52.0	0.454861	1.0
Combination 5: Opt + Ov	Combination 6: Opt + wght	39.0	0.807663	1.0

Como se observó que la técnica de oversampling obtenía sistemáticamente resultados muy inferiores a otras técnicas de balanceo de clases y, además, presentaba un tiempo de ejecución muy elevado, se optó por dejar de evaluar su rendimiento en modelos futuros.

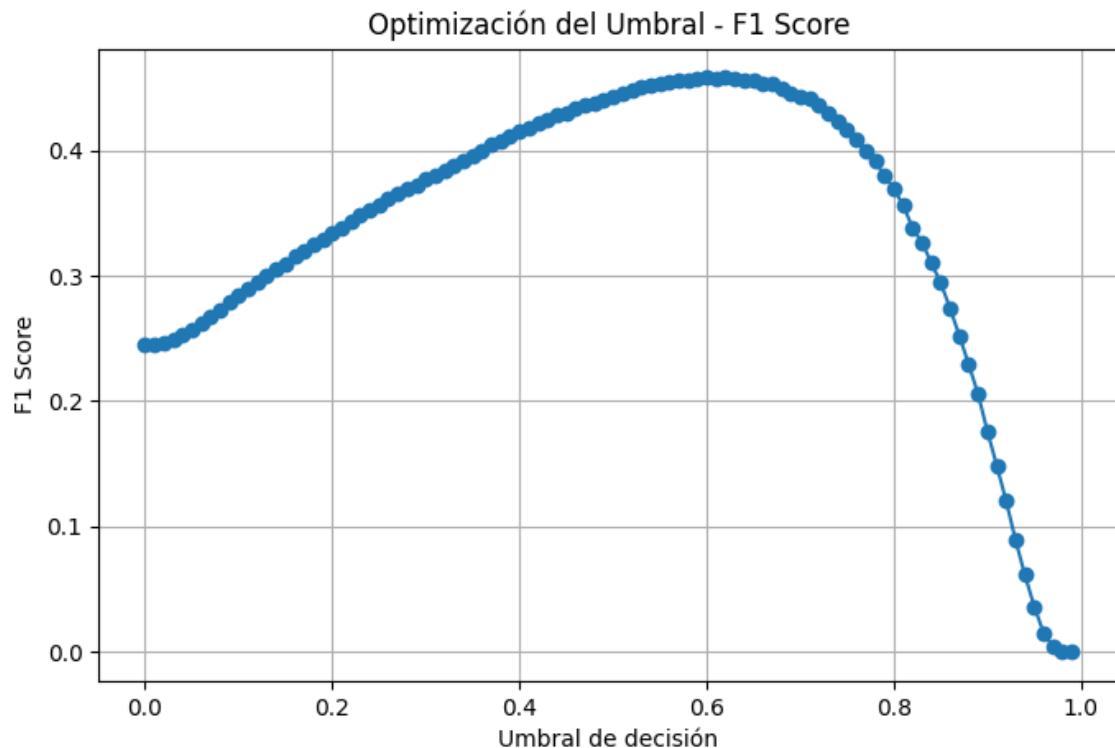
Por todo ello, se seleccionó como configuración óptima para el modelo final la Combinación 3, correspondiente a RandomizedSearchCV con class\_weight='balanced', al ser la que obtuvo el mayor AUC (0.820267) entre todas las variantes. Además, esta configuración ofrece un equilibrio convincente entre rendimiento predictivo (score mean elevado), estabilidad (baja desviación estándar de 0.0029) y un tiempo de ejecución razonable (5.59 minutos), lo que la convierte en una opción sólida para construir el modelo final.

### Final model

Una vez completadas las fases de ajuste de datos y de modelo, se construyó la versión definitiva de la regresión logística utilizando la configuración óptima de las fases 1 y 2: todas las variables y

`class_weight='balanced'`. En esta fase final no se repitió el proceso de búsqueda de hiperparámetros, ya que los valores óptimos fueron determinados previamente en la fase 2, por lo que se indicaron de forma manual.

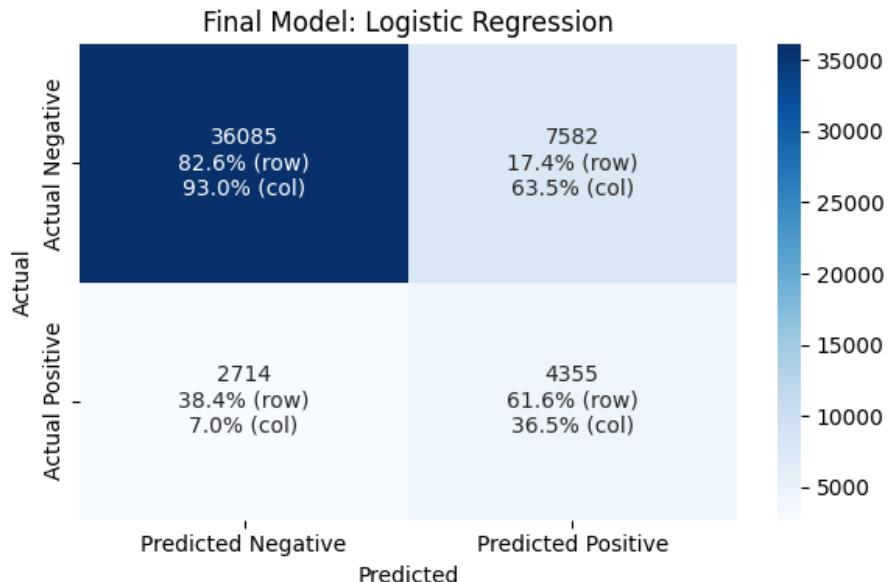
Para mejorar aún más el desempeño del modelo, se procedió a una optimización del umbral de decisión con el objetivo de maximizar el F1-score. Como se observa en la gráfica correspondiente, el umbral óptimo se encuentra alrededor de 0.6, punto en el que se maximiza el equilibrio entre precisión y sensibilidad.



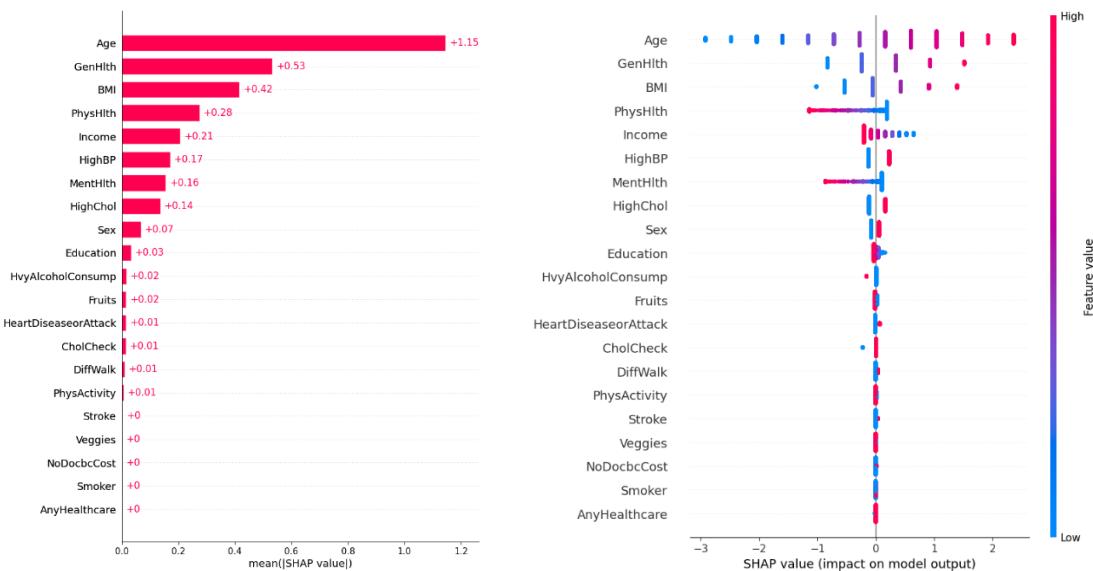
El rendimiento del modelo final se resume en la siguiente tabla de métricas: se alcanzó un AUC de 0.820, una exactitud global del 79.7% y una precisión del 36.5% y un recall del 61.6%. Estos resultados reflejan la capacidad del modelo para identificar correctamente la mayoría de los casos positivos, aunque con un porcentaje de falsos positivos aún relevante. La matriz de confusión confirma esta tendencia, mostrando que el modelo es sensible al diagnóstico positivo, aunque a costa de cierta pérdida de especificidad.

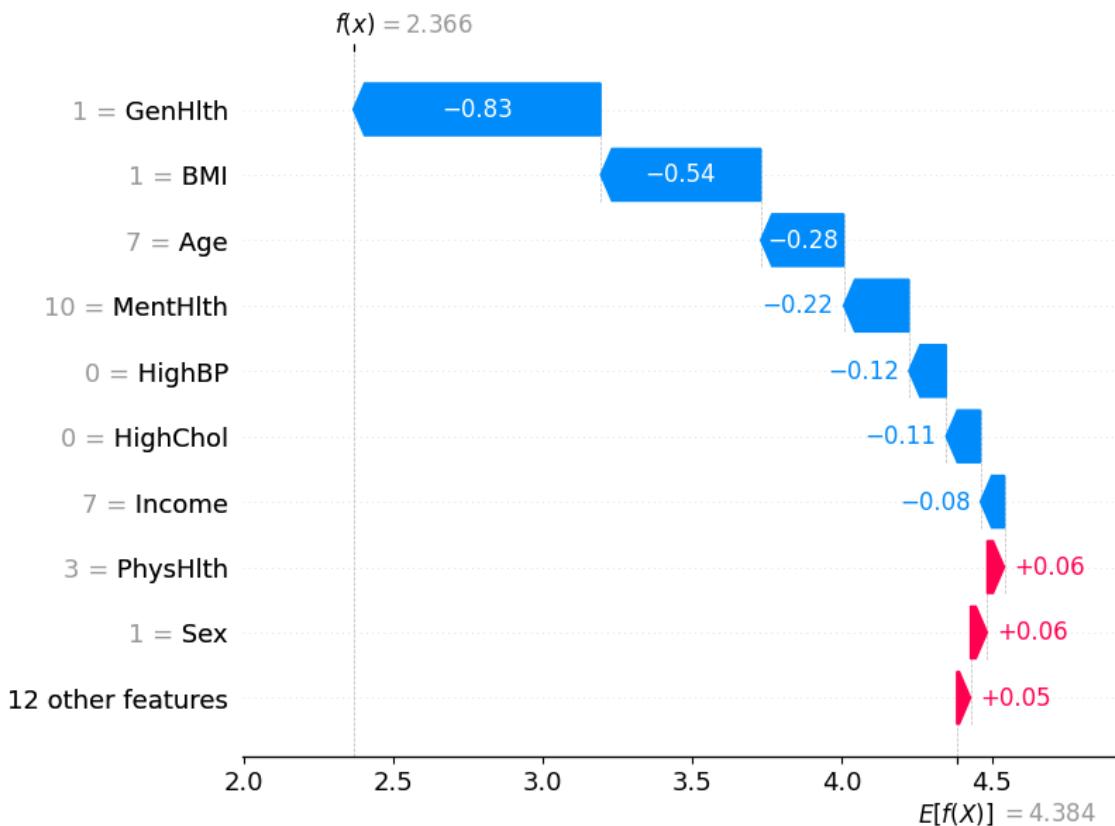
Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Final Model: Logistic Regression	0.820267	0.797067	0.61607	0.364832	0.824262	0.002896	0.92778



Desde la perspectiva de la interpretabilidad, se aplicaron técnicas basadas en SHAP para analizar la contribución de cada variable a las predicciones. El gráfico de resumen por barras indica que las variables más influyentes fueron Age, GenHlth, BMI y PhysHlth, todas ellas con una clara relevancia clínica en el contexto de la diabetes. El gráfico de dispersión (summary dot plot) permitió observar cómo ciertos valores extremos (como edades elevadas o peor estado general de salud) se asocian sistemáticamente con un aumento en la probabilidad de ser clasificado como prediabético o diabético. Finalmente, el gráfico tipo cascada (waterfall) explicó en detalle la decisión del modelo para un caso individual, mostrando cómo una combinación concreta de factores llevó a una predicción positiva.





### III.6.c.- Random Forest Classifier

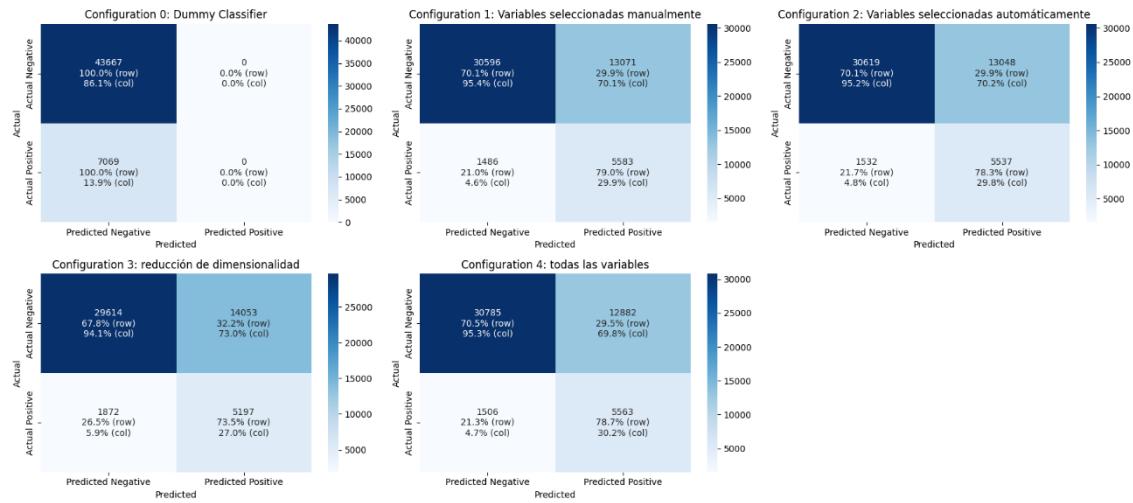
#### Fase 1

Durante la Fase 1 del ajuste del modelo Random Forest se exploraron distintas configuraciones de selección de variables con el objetivo de analizar su impacto sobre el rendimiento predictivo.

Las configuraciones que ofrecieron el mejor desempeño fueron la Configuración 4 (todas las variables) y la Configuración 1 (selección manual), con AUCs de 0.8205 y 0.8197 respectivamente, y F1-scores en torno a 0.824. Ambas superaron claramente al Dummy y mostraron una mejora notable respecto a la Configuración 2 (selección automática), que quedó ligeramente por detrás con un AUC de 0.814. Por otro lado, la Configuración 3 (reducción de dimensionalidad) ofreció un AUC sustancialmente más bajo (0.756), lo que sugiere que la eliminación de atributos mediante esta técnica condujo a una pérdida de información relevante para el modelo.

Summary Table:									
	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)	
0	Configuration 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000371	
1	Configuration 1: Variables seleccionadas manualmente	0.819767	0.713083	0.789786	0.299292	0.824423	0.002994	74.435281	
2	Configuration 2: Variables seleccionadas automáticamente	0.814340	0.712630	0.783279	0.297928	0.818250	0.004018	86.340021	
3	Configuration 3: reducción de dimensionalidad	0.756767	0.686120	0.735182	0.269974	0.756476	0.004526	541.599451	
4	Configuration 4: todas las variables	0.820522	0.716414	0.786957	0.301599	0.825368	0.003346	84.214120	

Las matrices de confusión refuerzan estas observaciones. Las configuraciones 1, 2 y 4 consiguieron recalls superiores al 78%, mostrando una buena capacidad para detectar casos positivos. La precisión, sin embargo, se mantuvo en niveles moderados (alrededor del 30%), consistente con la presencia de falsos positivos. En cambio, la Configuración 3 presentó un mayor número de errores en ambas clases, con un descenso en precisión (26.9%) y una caída en la exactitud global (68.6%), reflejando un menor equilibrio predictivo.



Desde el punto de vista estadístico, se aplicaron pruebas de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman confirmó diferencias globales significativas entre las configuraciones ( $p < 0.0001$ ). En las comparaciones por pares, el test de Wilcoxon señaló diferencias estadísticamente significativas entre la reducción de dimensionalidad y el resto de configuraciones, especialmente frente a la selección manual y el uso de todas las variables ( $p\text{-ajustado} < 0.01$ ). Estas diferencias también fueron detectadas mediante el test de Mann-Whitney U, reforzando la conclusión de que la reducción de atributos afectó negativamente al rendimiento del modelo Random Forest.

#### Resultados del Test de Friedman:

Estadístico	Friedman	p-value
	40.0	4.328423e-08

#### Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Configuration 0: Dummy Classifier	Configuration 1: man_sel	0.0	1.000000	1.000000
Configuration 0: Dummy Classifier	Configuration 2: aut_sel	0.0	1.000000	1.000000
Configuration 0: Dummy Classifier	Configuration 3: red_dim	0.0	1.000000	1.000000
Configuration 0: Dummy Classifier	Configuration 4: all	0.0	1.000000	1.000000
Configuration 1: man_sel	Configuration 2: aut_sel	55.0	0.000977	0.009766
Configuration 1: man_sel	Configuration 3: red_dim	55.0	0.000977	0.009766
Configuration 1: man_sel	Configuration 4: all	0.0	1.000000	1.000000
Configuration 2: aut_sel	Configuration 3: red_dim	55.0	0.000977	0.009766
Configuration 2: aut_sel	Configuration 4: all	0.0	1.000000	1.000000
Configuration 3: red_dim	Configuration 4: all	0.0	1.000000	1.000000

#### Resultados del Test de Kruskal-Wallis:

Estadístico	Kruskal-Wallis	p-value
	42.300329	1.445364e-08

#### Resultados del Test de Mann-Whitney U:

Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Configuration 0: Dummy Classifier	Configuration 1: man_sel	0.0	0.999977	1.000000
Configuration 0: Dummy Classifier	Configuration 2: aut_sel	0.0	0.999977	1.000000
Configuration 0: Dummy Classifier	Configuration 3: red_dim	0.0	0.999977	1.000000
Configuration 0: Dummy Classifier	Configuration 4: all	0.0	0.999977	1.000000
Configuration 1: man_sel	Configuration 2: aut_sel	87.0	0.002898	0.028977
Configuration 1: man_sel	Configuration 3: red_dim	100.0	0.000091	0.000913
Configuration 1: man_sel	Configuration 4: all	35.0	0.879339	1.000000
Configuration 2: aut_sel	Configuration 3: red_dim	100.0	0.000091	0.000913
Configuration 2: aut_sel	Configuration 4: all	11.0	0.998586	1.000000
Configuration 3: red_dim	Configuration 4: all	0.0	0.999933	1.000000

Por todo ello, se seleccionó como configuración óptima para la siguiente fase de ajuste la Configuración 4, que emplea todas las variables originales, al ofrecer un equilibrio ideal entre rendimiento predictivo (AUC alto y score mean elevado), estabilidad (desviación estándar de 0.0033) y eficiencia computacional razonable para un modelo de este tipo (84.2 minutos de entrenamiento).

## Fase 2

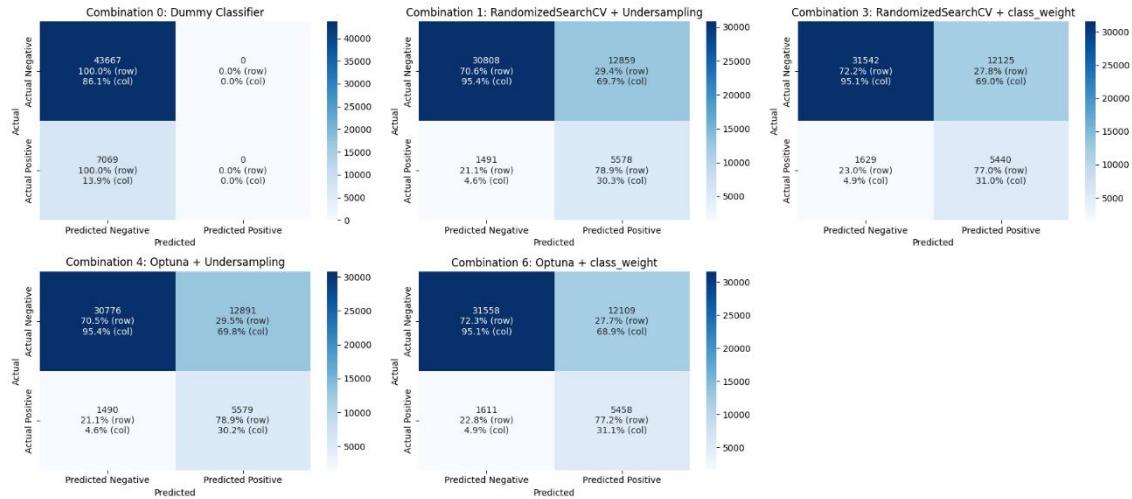
Durante la Fase 2 se evaluaron distintas estrategias de ajuste fino del modelo Random Forest, combinando dos técnicas de optimización de hiperparámetros (RandomizedSearchCV y Optuna) con tres métodos de balanceo de clases: undersampling, class\_weight y su combinación con optimización bayesiana. El objetivo fue determinar si alguna de estas combinaciones mejoraba de forma significativa el rendimiento observado en la Fase 1.

Los resultados mostraron un rendimiento muy homogéneo entre las configuraciones, con valores de AUC comprendidos entre 0.8207 y 0.8213. La combinación que alcanzó el mayor AUC fue la Combinación 6 (Optuna + class\_weight), con un valor de 0.821267. Aunque las diferencias en rendimiento fueron pequeñas, esta combinación se destacó también por ofrecer el mayor nivel de precisión (0.3107) sin comprometer el recall (77.2%).

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Combination 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000884
1	Combination 1: RandomizedSearchCV + Undersampling	0.820725	0.717163	0.789079	0.302544	0.825594	0.003329	84.210967
2	Combination 3: RandomizedSearchCV + class_weight	0.821103	0.728910	0.769557	0.309707	0.825932	0.003528	363.538219
3	Combination 4: Optuna + Undersampling	0.820688	0.716552	0.789221	0.302057	0.825723	0.003292	77.100125
4	Combination 6: Optuna + class_weight	0.821267	0.729581	0.772104	0.310696	0.826084	0.003460	242.413572

Las matrices de confusión corroboran estos resultados: todas las combinaciones (1 a 4) mostraron un recall superior al 76.9%, con valores de precisión entre el 30% y el 31%, lo que refleja una buena capacidad de detección de la clase positiva, aunque con una proporción considerable de falsos positivos. Estas cifras son representativas de modelos bien ajustados en contextos de desbalanceo de clases.



Desde el punto de vista estadístico, el test de Friedman confirmó la existencia de diferencias globales significativas entre las combinaciones ( $p < 0.0001$ ). No obstante, los tests por pares (Wilcoxon y Mann-Whitney U) no mostraron diferencias estadísticamente significativas entre las combinaciones más competitivas, lo cual sugiere que el rendimiento es estable independientemente de la técnica de balanceo empleada, siempre que esté correctamente ajustada. Algunas diferencias puntuales, como entre class\_weight y undersampling, se manifestaron en los valores de AUC y score mean, pero no alcanzaron significación estadística tras la corrección por comparaciones múltiples.

Resultados del Test de Friedman:

Estadístico Friedman	p-value
24.08	0.000077

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	1.000000	1.0
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	1.000000	1.0
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	1.000000	1.0
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	1.000000	1.0
Combination 1: RandCV + Un	Combination 3: RandCV + wght	13.0	0.934570	1.0
Combination 1: RandCV + Un	Combination 4: Opt + Un	2.0	0.998047	1.0
Combination 1: RandCV + Un	Combination 6: Opt + wght	12.0	0.947266	1.0
Combination 3: RandCV + wght	Combination 4: Opt + Un	37.0	0.187500	1.0
Combination 3: RandCV + wght	Combination 6: Opt + wght	13.0	0.934570	1.0
Combination 4: Opt + Un	Combination 6: Opt + wght	13.0	0.934570	1.0

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis	p-value
23.974424	0.000081

Resultados del Test de Mann-Whitney U:

Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	0.999977	1.0
Combination 1: RandCV + Un	Combination 3: RandCV + wght	45.0	0.661208	1.0
Combination 1: RandCV + Un	Combination 4: Opt + Un	46.0	0.633135	1.0
Combination 1: RandCV + Un	Combination 6: Opt + wght	44.0	0.688412	1.0
Combination 3: RandCV + wght	Combination 4: Opt + Un	54.0	0.395668	1.0
Combination 3: RandCV + wght	Combination 6: Opt + wght	46.0	0.633135	1.0
Combination 4: Opt + Un	Combination 6: Opt + wght	44.0	0.688412	1.0

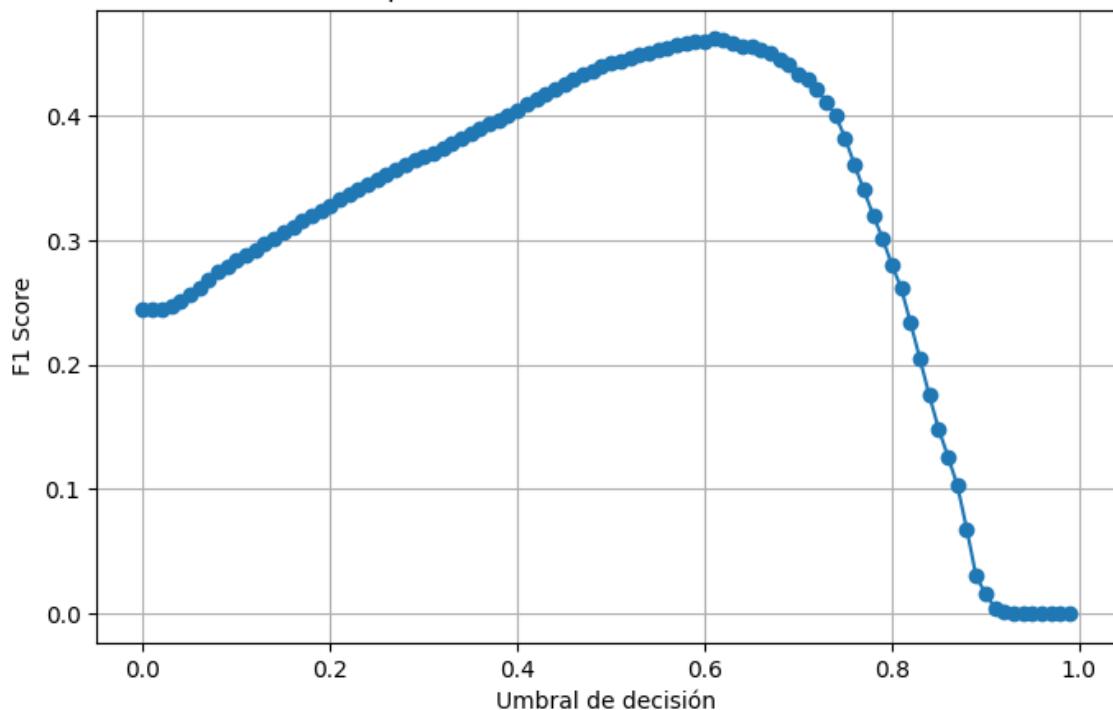
Por todo ello, se seleccionó como configuración óptima para el modelo final la Combinación 6, correspondiente a Optuna con `class_weight='balanced'`, al ser la que obtuvo el mayor AUC (0.821267) entre todas las variantes. Esta configuración ofrece además un buen equilibrio entre rendimiento predictivo, estabilidad (desviación estándar de 0.0035) y una razonable eficiencia computacional dentro de los estándares del modelo (tiempo de entrenamiento: 242 minutos).

### *Final model*

Una vez completadas las fases de ajuste de datos y de modelo, se construyó la versión definitiva del modelo Random Forest utilizando la configuración óptima identificada: todas las variables con `class_weight='balanced'`. Esta combinación, seleccionada por haber obtenido el mayor AUC en la Fase 2 (0.821267), ofrecía un equilibrio ideal entre rendimiento predictivo, estabilidad y coste computacional razonable.

Durante esta fase no se realizó una nueva búsqueda de hiperparámetros, ya que los valores óptimos fueron determinados previamente y se añadieron manualmente al modelo final. En su lugar, se procedió a optimizar el umbral de decisión para maximizar el F1-score. Tal como muestra la gráfica correspondiente, el umbral óptimo se situó aproximadamente en 0.6, lo que permitió mejorar la armonización entre precisión y sensibilidad del modelo.

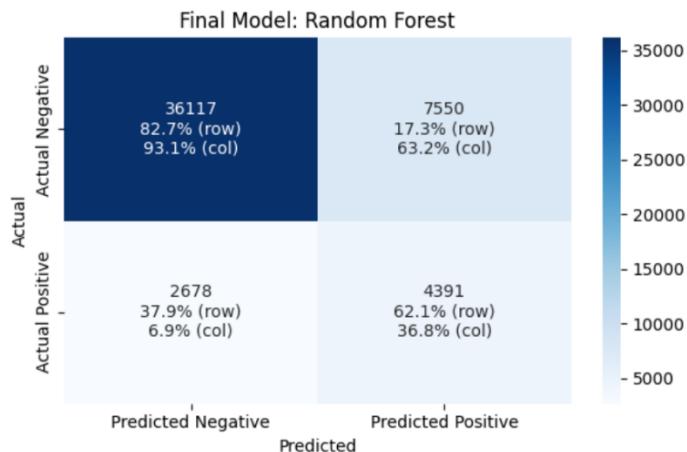
### Optimización del Umbral - F1 Score



El rendimiento del modelo final se resume en la siguiente tabla: se alcanzó un AUC de 0.821, una exactitud global del 79.8%, un recall del 62.1% y una precisión del 36.8%. La matriz de confusión refleja esta buena capacidad para identificar casos positivos, manteniendo una tasa aceptable de falsos positivos, lo que resulta particularmente relevante en contextos de desbalance de clases.

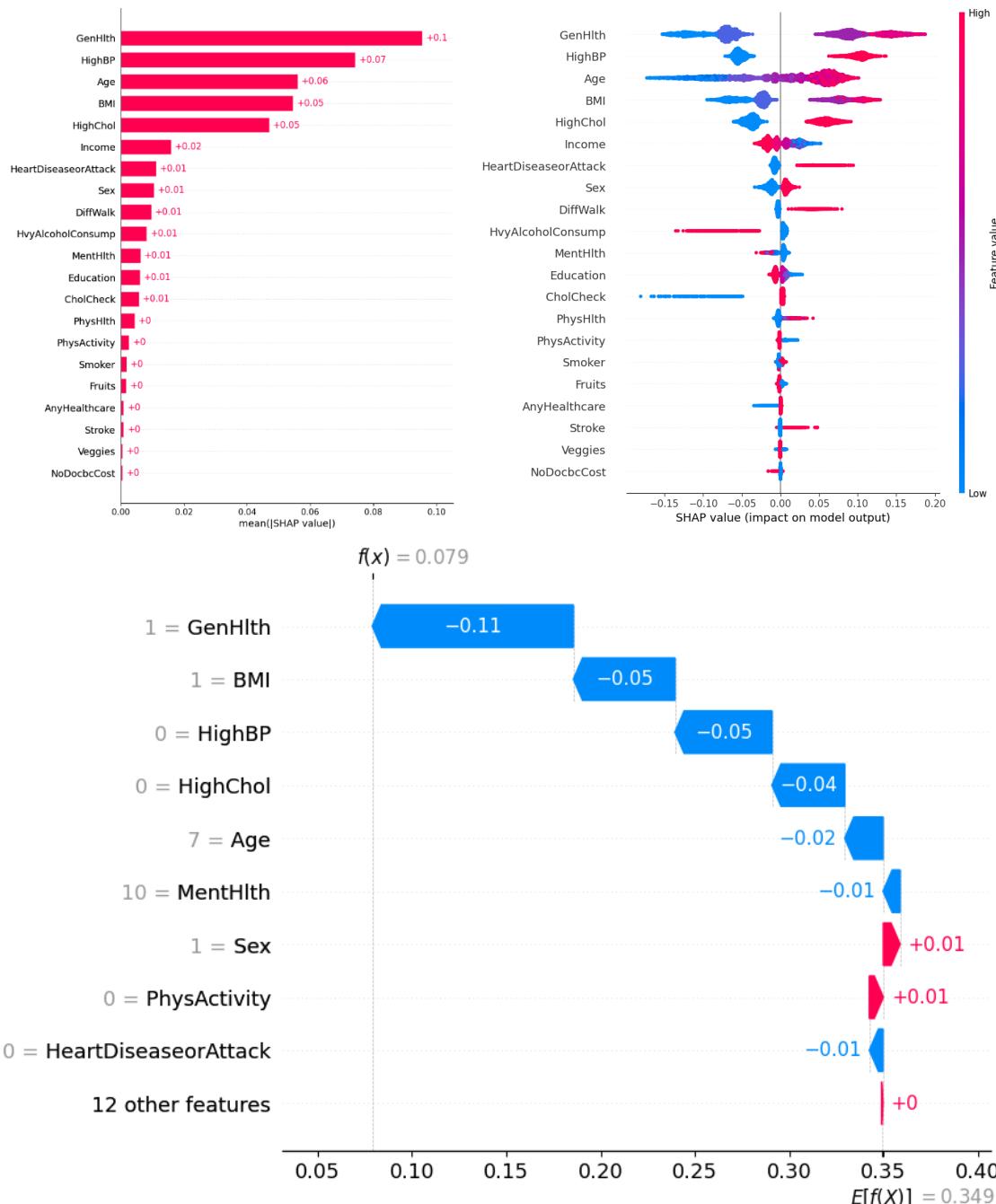
Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Final Model: Random Forest	0.821267	0.798407	0.621163	0.367725	0.826084	0.00346	0.876639



Desde el punto de vista de la interpretabilidad, se emplearon valores SHAP para analizar la contribución individual de cada variable en las predicciones. El gráfico de resumen por barras destaca a GenHlth, HighBP, Age, BMI y HighChol como las variables con mayor peso en la decisión del modelo, todas ellas con una clara conexión clínica con la diabetes. En el gráfico de dispersión (summary dot plot) se observó que los peores niveles de salud general, hipertensión y colesterol elevado se asociaron con un incremento notable en la probabilidad de clasificación positiva. Por último, el

gráfico waterfall mostró cómo una combinación específica de variables llevó a una predicción positiva en un caso individual, detallando el efecto marginal de cada atributo sobre la probabilidad final.



### III.6.d.- Balanced Random Forest Classifier

#### Fase 1

Durante la Fase 1 del ajuste del modelo Balanced Random Forest se evaluó el impacto de distintas estrategias de selección de variables sobre el rendimiento predictivo.

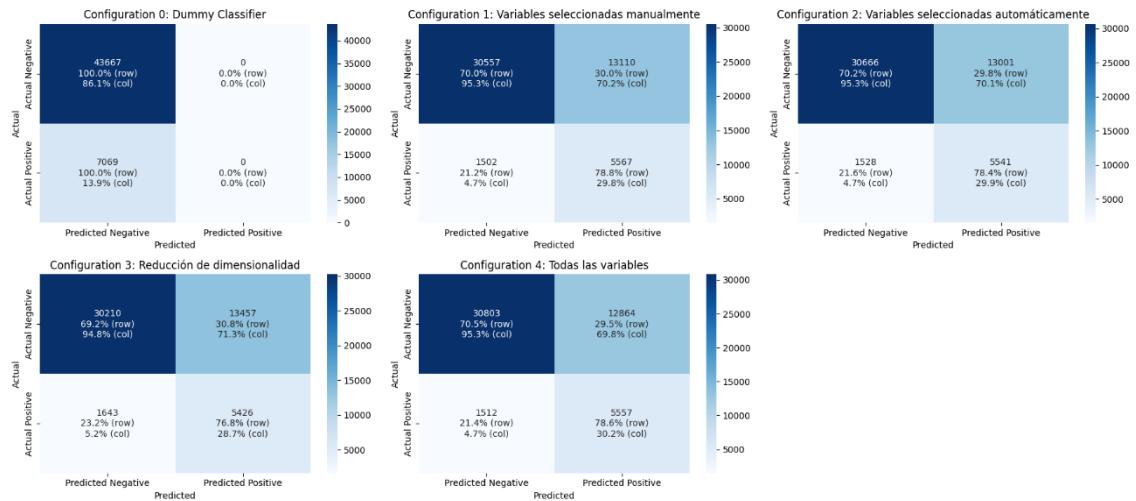
Las mejores configuraciones fueron la Configuración 4 (todas las variables) y la Configuración 1 (selección manual), con AUCs de 0.8205 y 0.8197 respectivamente, y puntuaciones medias por encima de 0.824, lo que demuestra una elevada capacidad discriminativa del modelo. La

Configuración 2 (selección automática), aunque competitiva, quedó ligeramente por debajo con un AUC de 0.817. Por su parte, la Configuración 3 (reducción de dimensionalidad) mostró un rendimiento considerablemente inferior, alcanzando un AUC de solo 0.743, lo que sugiere pérdida de información valiosa durante el proceso de compresión de atributos.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Configuration 0: Dummy Classifier	0.50000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000884
1	Configuration 1: Variables seleccionadas manualmente	0.819656	0.711999	0.787523	0.298067	0.824228	0.003068	70.422371
2	Configuration 2: Variables seleccionadas automáticamente	0.817093	0.713635	0.783845	0.298835	0.821371	0.003668	107.319923
3	Configuration 3: Reducción de dimensionalidad	0.743059	0.702381	0.767577	0.287348	0.752877	0.004382	450.477261
4	Configuration 4: Todas las variables	0.820543	0.716651	0.786108	0.301667	0.825508	0.003394	83.271273

Las matrices de confusión muestran que las configuraciones 1, 2 y 4 lograron recalls superiores al 78%, con una precisión ligeramente superior al 29%, reflejando un buen equilibrio entre sensibilidad y especificidad en el contexto de clases desbalanceadas. En cambio, la Configuración 3 mostró una mayor tasa de falsos positivos, con una precisión reducida del 28.7% y una leve caída en la exactitud global (70.2%), lo que indica un deterioro en la calidad general del modelo.



Desde el punto de vista estadístico, se aplicaron las pruebas de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman reveló diferencias globales significativas entre las configuraciones ( $p < 0.0001$ ), mientras que los tests por pares indicaron que la reducción de dimensionalidad se comportó de forma significativamente distinta a las demás configuraciones, como lo muestran los resultados del test de Wilcoxon ( $p$ -ajustado  $< 0.01$ ) y del Mann-Whitney U, especialmente en comparación con la selección automática y con el uso de todas las variables.

```

Resultados del Test de Friedman:
Estadístico Friedman      p-value
        40.0 4.328423e-08

Resultados del Test de Wilcoxon:
          Primer Modelo      Segundo Modelo  Wilcox V  p-value original  p-value ajustado
Configuration 0: Dummy Classifier Configuration 1: man_sel    0.0      1.000000  1.000000
Configuration 0: Dummy Classifier Configuration 2: aut_sel    0.0      1.000000  1.000000
Configuration 0: Dummy Classifier Configuration 3: red_dim    0.0      1.000000  1.000000
Configuration 0: Dummy Classifier Configuration 4: all       0.0      1.000000  1.000000
Configuration 1: man_sel Configuration 2: aut_sel    55.0     0.000977  0.009766
Configuration 1: man_sel Configuration 3: red_dim    55.0     0.000977  0.009766
Configuration 1: man_sel Configuration 4: all       0.0      1.000000  1.000000
Configuration 2: aut_sel Configuration 3: red_dim    55.0     0.000977  0.009766
Configuration 2: aut_sel Configuration 4: all       0.0      1.000000  1.000000
Configuration 3: red_dim Configuration 4: all       0.0      1.000000  1.000000

Resultados del Test de Kruskal-Wallis:
Estadístico Kruskal-Wallis      p-value
        40.620194 3.221235e-08

Resultados del Test de Mann-Whitney U:
          Primer Modelo      Segundo Modelo      U  p-value original  p-value ajustado
Configuration 0: Dummy Classifier Configuration 1: man_sel    0.0      0.999977  1.000000
Configuration 0: Dummy Classifier Configuration 2: aut_sel    0.0      0.999977  1.000000
Configuration 0: Dummy Classifier Configuration 3: red_dim    0.0      0.999977  1.000000
Configuration 0: Dummy Classifier Configuration 4: all       0.0      0.999977  1.000000
Configuration 1: man_sel Configuration 2: aut_sel    77.0     0.022577  0.225773
Configuration 1: man_sel Configuration 3: red_dim 100.0     0.000091  0.000913
Configuration 1: man_sel Configuration 4: all      32.0     0.919014  1.000000
Configuration 2: aut_sel Configuration 3: red_dim 100.0     0.000091  0.000913
Configuration 2: aut_sel Configuration 4: all      20.0     0.989433  1.000000
Configuration 3: red_dim Configuration 4: all      0.0      0.999933  1.000000

```

Por todo ello, se seleccionó como configuración óptima para la siguiente fase de ajuste la Configuración 4, que emplea todas las variables originales, al ofrecer un equilibrio ideal entre rendimiento predictivo (AUC alto y F1-score elevado), estabilidad (desviación estándar moderada) y una eficiencia computacional aceptable (83.27 minutos), especialmente tratándose de un algoritmo basado en conjuntos de árboles balanceados.

## Fase 2

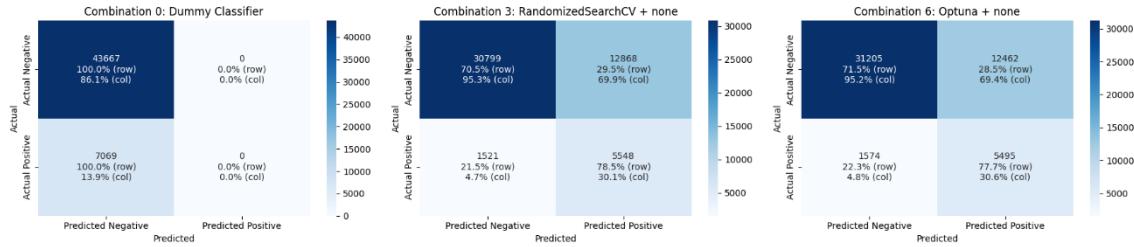
Durante la Fase 2 se evaluó el impacto de dos técnicas de ajuste fino aplicadas al modelo Balanced Random Forest: RandomizedSearchCV y Optuna, ambas sin aplicar técnicas adicionales de balanceo, dado que el algoritmo ya incorpora internamente mecanismos para equilibrar las clases durante el entrenamiento y podría resultar contraproducente realizar una técnica de balanceo adicional. El objetivo era comprobar si alguna estrategia de optimización podía aportar mejoras marginales al rendimiento obtenido en la Fase 1.

Los resultados mostraron un rendimiento muy uniforme entre ambas combinaciones, con valores de AUC de 0.8221 para RandomizedSearchCV y 0.8225 para Optuna. La Combinación 6 (Optuna sin técnicas adicionales) fue la que obtuvo el mayor AUC, además de requerir menos tiempo de ejecución (75.93 minutos), lo que refuerza su eficiencia sin comprometer la calidad predictiva.

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Combination 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000786
1	Combination 3: RandomizedSearchCV + none	0.822117	0.716395	0.784835	0.301260	0.826697	0.003361	109.185137
2	Combination 6: Optuna + none	0.822478	0.723352	0.777338	0.306009	0.827014	0.003395	75.925826

Las matrices de confusión muestran un comportamiento casi idéntico entre ambas variantes: ambos modelos lograron identificar correctamente alrededor del 77.7–78.4% de los casos positivos, con precisiones cercanas al 30%, evidenciando un equilibrio razonable entre sensibilidad y especificidad.

Estas cifras son consistentes con las obtenidas en la fase anterior, lo que sugiere estabilidad del modelo frente a ligeras variaciones en la técnica de optimización.



A nivel estadístico, el test de Friedman confirmó la existencia de diferencias globales entre los modelos ( $p < 0.0003$ ), pero los análisis por pares (Wilcoxon y Mann-Whitney U) no revelaron diferencias significativas entre RandomizedSearchCV y Optuna, lo que implica que ambas combinaciones pueden considerarse equivalentes en términos de rendimiento.

Resultados del Test de Friedman:

Estadístico Friedman	p-value
16.8	0.000225

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 3: RandCV + none	0.0	1.000000	1.0
Combination 0: Dummy Classifier	Combination 6: Opt + none	0.0	1.000000	1.0
Combination 3: RandCV + none	Combination 6: Opt + none	5.0	0.993164	1.0

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis	p-value
20.135242	0.000042

Resultados del Test de Mann-Whitney U:

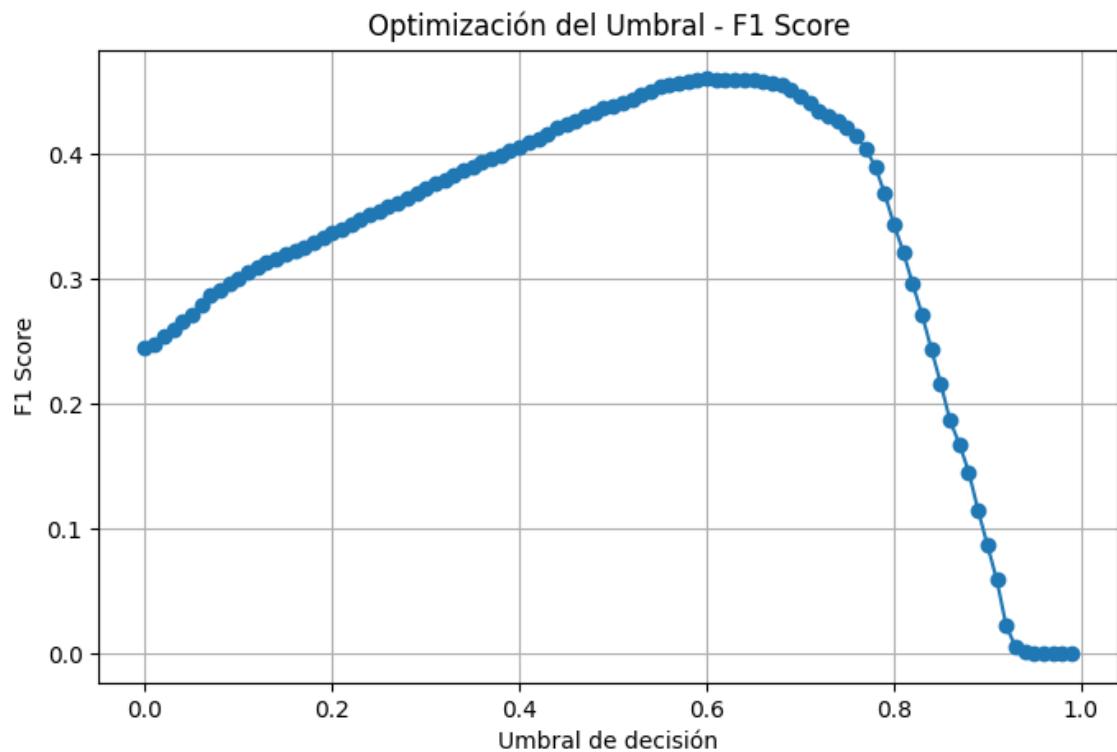
Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 3: RandCV + none	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 6: Opt + none	0.0	0.999977	1.0
Combination 3: RandCV + none	Combination 6: Opt + none	46.0	0.633135	1.0

La ligera superioridad de la configuración 6 en precisión y eficiencia computacional, no obstante, justifica su elección como variante óptima.

### Final model

Una vez completadas las fases de ajuste de datos y de modelo, se construyó la versión definitiva del modelo Balanced Random Forest utilizando la configuración óptima identificada: todas las variables y sin técnicas adicionales de balanceo adicionales. Esta combinación fue seleccionada por haber obtenido el mayor AUC en la Fase 2 (0.8225), con una precisión y estabilidad competitivas, y un tiempo de ejecución inferior al de otras configuraciones.

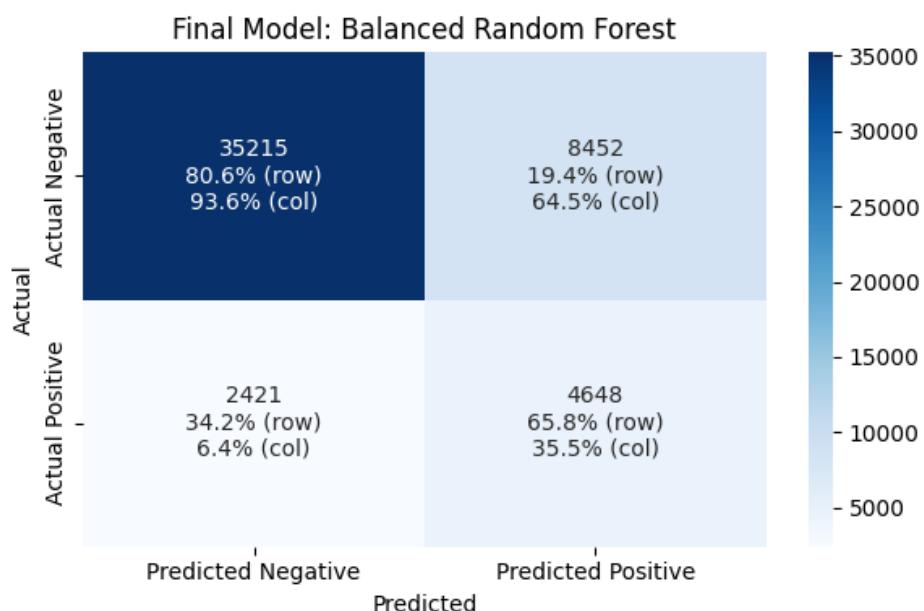
En esta fase final no se volvió a ejecutar la búsqueda de hiperparámetros, ya que los valores óptimos habían sido determinados previamente. Se procedió, en cambio, a una optimización del umbral de decisión con el fin de maximizar el F1-score, que alcanzó su valor máximo en torno a un umbral de 0.6, punto que representa el mejor compromiso entre sensibilidad y precisión.



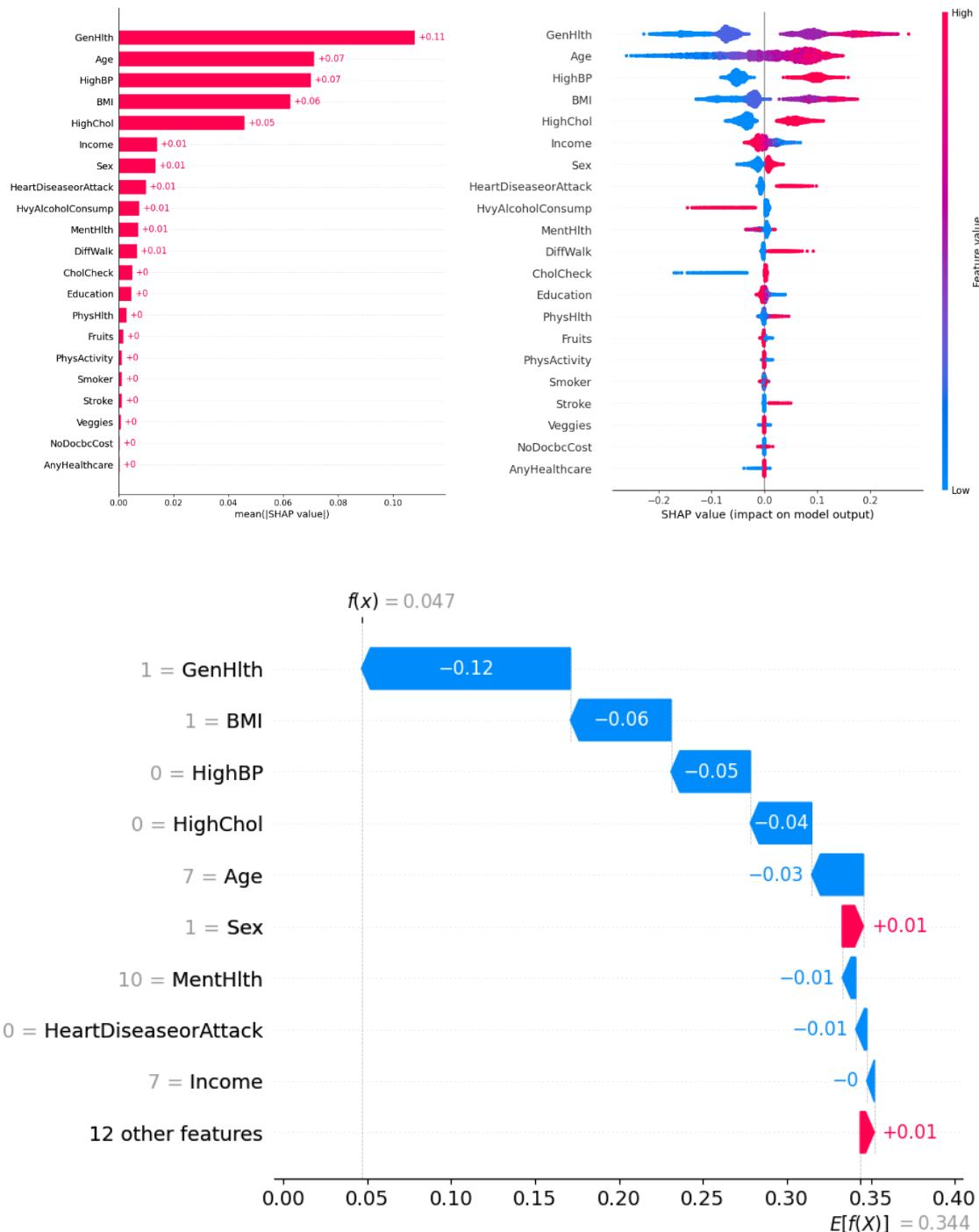
El modelo final alcanzó un AUC de 0.8225, una exactitud del 78.6%, y un F1-score promedio de 0.8270, con una precisión del 35.5% y un recall del 65.8%. Estos resultados reflejan un buen rendimiento global, destacando especialmente en la detección de casos positivos, lo cual es esencial en un contexto clínico como el de la detección de diabetes. La matriz de confusión muestra un equilibrio razonable entre verdaderos positivos y falsos positivos, acorde con la naturaleza desbalanceada del problema.

Summary Table:

Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0 Final Model: Balanced Random Forest	0.822478	0.785695	0.657519	0.354809	0.827014	0.003395	0.484018



Desde la perspectiva de la interpretabilidad, se emplearon valores SHAP para evaluar la contribución individual de las variables. El gráfico de resumen por barras identifica como atributos más influyentes a GenHlth, Age, HighBP, BMI y HighChol, todos ellos con gran relevancia clínica. El gráfico de dispersión muestra que peores niveles de salud general, hipertensión y edad avanzada están asociados a un mayor riesgo de ser clasificado como positivo. Finalmente, el gráfico tipo cascada ilustra el razonamiento del modelo en una predicción concreta, revelando cómo las variables más influyentes se combinan para superar el umbral de clasificación.



### III.6.e.- LightGBM

#### Fase 1

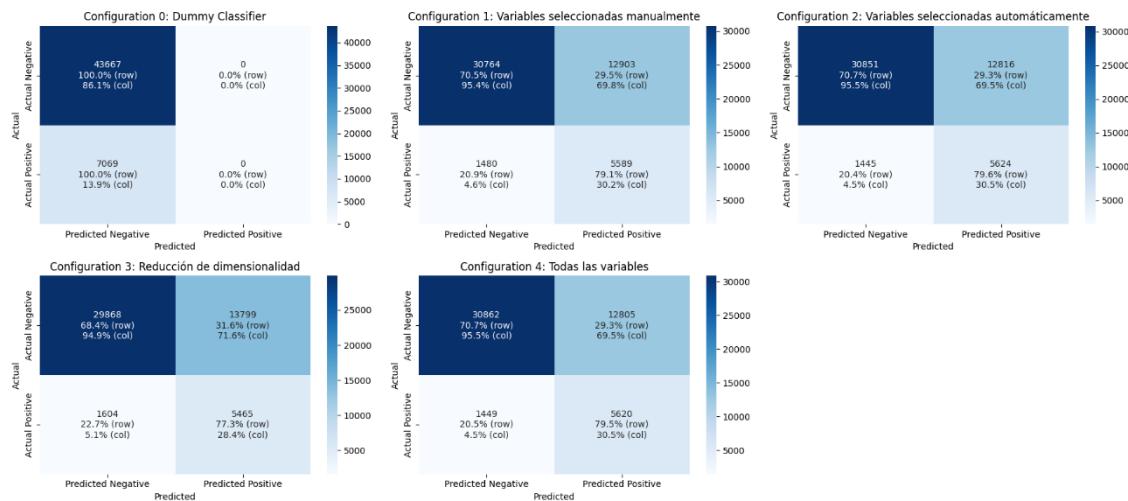
Durante la Fase 1 del ajuste del modelo LightGBM se exploraron distintas estrategias de selección de variables con el fin de evaluar su impacto sobre el rendimiento predictivo del modelo.

Los mejores resultados se obtuvieron con la Configuración 2 (selección automática), que alcanzó el AUC más alto (0.825350), seguida muy de cerca por la Configuración 4 (todas las variables) con AUC de 0.8249. Ambas configuraciones demostraron una capacidad predictiva elevada y estable. La Configuración 1 (selección manual) también fue competitiva (AUC = 0.823071), aunque presentó una ligera disminución en precisión. Por otro lado, la Configuración 3 (reducción de dimensionalidad) mostró un descenso notable en el rendimiento (AUC = 0.742699), indicando pérdida de información relevante.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Configuration 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000585
1	Configuration 1: Variables seleccionadas manualmente	0.823071	0.716513	0.790635	0.302239	0.826968	0.003197	23.944999
2	Configuration 2: Variables seleccionadas automáticamente	0.825350	0.718918	0.795586	0.304989	0.829133	0.003344	33.076377
3	Configuration 3: Reducción de dimensionalidad	0.742699	0.696409	0.773094	0.283690	0.753163	0.005205	370.730903
4	Configuration 4: Todas las variables	0.824886	0.719056	0.795021	0.305020	0.828728	0.003189	25.251415

Las matrices de confusión confirman estas observaciones. Las configuraciones 1, 2 y 4 muestran un recall cercano al 79.5%, con precisiones de entre 30.2% y 30.5%, reflejando un buen equilibrio en un contexto con clases desbalanceadas. En contraste, la Configuración 3 mostró un mayor número de falsos positivos y una precisión reducida (28.4%), junto con una menor exactitud (69.6%).



Desde un punto de vista estadístico, los resultados fueron analizados mediante pruebas de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman confirmó la existencia de diferencias significativas entre configuraciones ( $p < 0.0001$ ), y el test de Wilcoxon detectó diferencias significativas entre la Configuración 3 y el resto ( $p\text{-ajustado} < 0.01$ ), especialmente frente a las configuraciones que conservaban mayor cantidad de información, como la selección automática y el uso de todas las variables.

Resultados del Test de Friedman:  
 Estadístico Friedman p-value  
 38.72 7.958599e-08

Resultados del Test de Wilcoxon:  
 Primer Modelo Segundo Modelo Wilcox V p-value original p-value ajustado  
 Configuration 0: Dummy Classifier Configuration 1: man\_sel 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 2: aut\_sel 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 3: red\_dim 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 4: all 0.0 1.000000 1.000000  
 Configuration 1: man\_sel Configuration 2: aut\_sel 0.0 1.000000 1.000000  
 Configuration 1: man\_sel Configuration 3: red\_dim 55.0 0.000977 0.009766  
 Configuration 1: man\_sel Configuration 4: all 0.0 1.000000 1.000000  
 Configuration 2: aut\_sel Configuration 3: red\_dim 55.0 0.000977 0.009766  
 Configuration 2: aut\_sel Configuration 4: all 51.0 0.006836 0.068359  
 Configuration 3: red\_dim Configuration 4: all 0.0 1.000000 1.000000

Resultados del Test de Kruskal-Wallis:  
 Estadístico Kruskal-Wallis p-value  
 39.663911 5.079503e-08

Resultados del Test de Mann-Whitney U:  
 Primer Modelo Segundo Modelo U p-value original p-value ajustado  
 Configuration 0: Dummy Classifier Configuration 1: man\_sel 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 2: aut\_sel 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 3: red\_dim 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 4: all 0.0 0.999977 1.000000  
 Configuration 1: man\_sel Configuration 2: aut\_sel 23.0 0.981182 1.000000  
 Configuration 1: man\_sel Configuration 3: red\_dim 100.0 0.000091 0.000913  
 Configuration 1: man\_sel Configuration 4: all 29.0 0.947945 1.000000  
 Configuration 2: aut\_sel Configuration 3: red\_dim 100.0 0.000091 0.000913  
 Configuration 2: aut\_sel Configuration 4: all 56.0 0.338792 1.000000  
 Configuration 3: red\_dim Configuration 4: all 0.0 0.999933 1.000000

Por todo ello, se seleccionó como configuración óptima para la siguiente fase de ajuste la Configuración 2, correspondiente a la selección automática de variables, al ser la que ofrece el mayor rendimiento predictivo (AUC más alto), una excelente estabilidad (score std: 0.0033) y un tiempo de ejecución razonable de 33.08 minutos.

## Fase 2

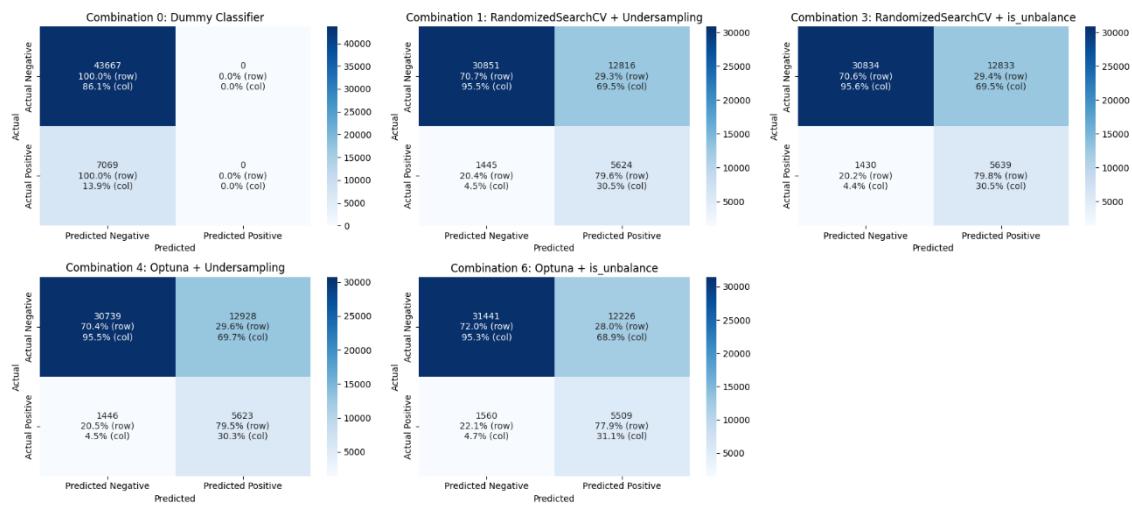
Durante la Fase 2 se analizaron distintas combinaciones de optimización de hiperparámetros y técnicas de balanceo para afinar el rendimiento del modelo LightGBM. En esta ocasión, se emplearon tanto RandomizedSearchCV como Optuna en combinación con dos estrategias de balanceo: undersampling y la activación del parámetro is\_unbalance=True, específico de este modelo. El objetivo fue determinar si alguna combinación aportaba mejoras respecto a la configuración óptima obtenida en la fase anterior.

Los resultados mostraron diferencias sutiles pero consistentes. La Combinación 3 (RandomizedSearchCV + is\_unbalance) obtuvo el mayor AUC (0.826039), convirtiéndose en la mejor candidata para el modelo final. Le siguieron muy de cerca la Combinación 6 (Optuna + is\_unbalance) y la Combinación 1 (RandomizedSearchCV + undersampling), ambas con un AUC de aproximadamente 0.8252 y un rendimiento muy estable (score std entre 0.0033 y 0.0035). En contraste, aunque las variantes con undersampling ofrecieron buena capacidad predictiva, no lograron superar el rendimiento de las combinaciones con is\_unbalance, especialmente en términos de AUC.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Combination 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.002863
1	Combination 1: RandomizedSearchCV + Undersampling	0.825350	0.718918	0.795586	0.304989	0.829133	0.003344	101.442556
2	Combination 3: RandomizedSearchCV + is_unbalance	0.826039	0.718878	0.797708	0.305273	0.829994	0.003525	204.995117
3	Combination 4: Optuna + Undersampling	0.824186	0.716690	0.795445	0.303110	0.827543	0.003203	18.794506
4	Combination 6: Optuna + is_unbalance	0.825176	0.728280	0.779318	0.310629	0.829134	0.003334	50.742566

Las matrices de confusión refuerzan estas conclusiones. Todas las combinaciones útiles (1 a 4) alcanzaron un recall de entre 77.9% y 79.7%, junto con precisiones en torno al 30.5%, reflejando una respuesta equilibrada frente al problema de desbalance de clases. La Combinación 3 en particular mantuvo un excelente equilibrio entre sensibilidad (recall) y especificidad, sin aumentar de forma drástica la tasa de falsos positivos.



Desde el punto de vista estadístico, el test de Friedman detectó diferencias significativas entre combinaciones ( $p < 0.0001$ ). El test de Wilcoxon también confirmó estas diferencias, especialmente entre combinaciones con y sin el parámetro `is_unbalance`, siendo la Combinación 3 significativamente mejor que la Combinación 4 (Optuna + undersampling) ( $p$ -ajustado  $< 0.01$ ). Estas observaciones se vieron apoyadas por los resultados del test de Kruskal-Wallis y el test de Mann-Whitney U, aunque este último mostró diferencias menos consistentes entre pares.

Resultados del Test de Friedman:  
 Estadístico Friedman p-value  
 38.0 1.120559e-07

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	1.000000	1.000000
Combination 1: RandCV + Un	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 1: RandCV + Un	Combination 4: Opt + Un	55.0	0.000977	0.009766
Combination 1: RandCV + Un	Combination 6: Opt + wght	27.0	0.539062	1.000000
Combination 3: RandCV + wght	Combination 4: Opt + Un	55.0	0.000977	0.009766
Combination 3: RandCV + wght	Combination 6: Opt + wght	55.0	0.000977	0.009766
Combination 4: Opt + Un	Combination 6: Opt + wght	0.0	1.000000	1.000000

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis p-value  
 27.649661 0.000015

Resultados del Test de Mann-Whitney U:

Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	0.999977	1.000000
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	0.999977	1.000000
Combination 0: Dummy Classifier	Combination 4: Opt + Un	0.0	0.999977	1.000000
Combination 0: Dummy Classifier	Combination 6: Opt + wght	0.0	0.999977	1.000000
Combination 1: RandCV + Un	Combination 3: RandCV + wght	35.0	0.879339	1.000000
Combination 1: RandCV + Un	Combination 4: Opt + Un	72.0	0.052055	0.520549
Combination 1: RandCV + Un	Combination 6: Opt + wght	51.0	0.484925	1.000000
Combination 3: RandCV + wght	Combination 4: Opt + Un	78.0	0.018818	0.188177
Combination 3: RandCV + wght	Combination 6: Opt + wght	63.0	0.172352	1.000000
Combination 4: Opt + Un	Combination 6: Opt + wght	29.0	0.947945	1.000000

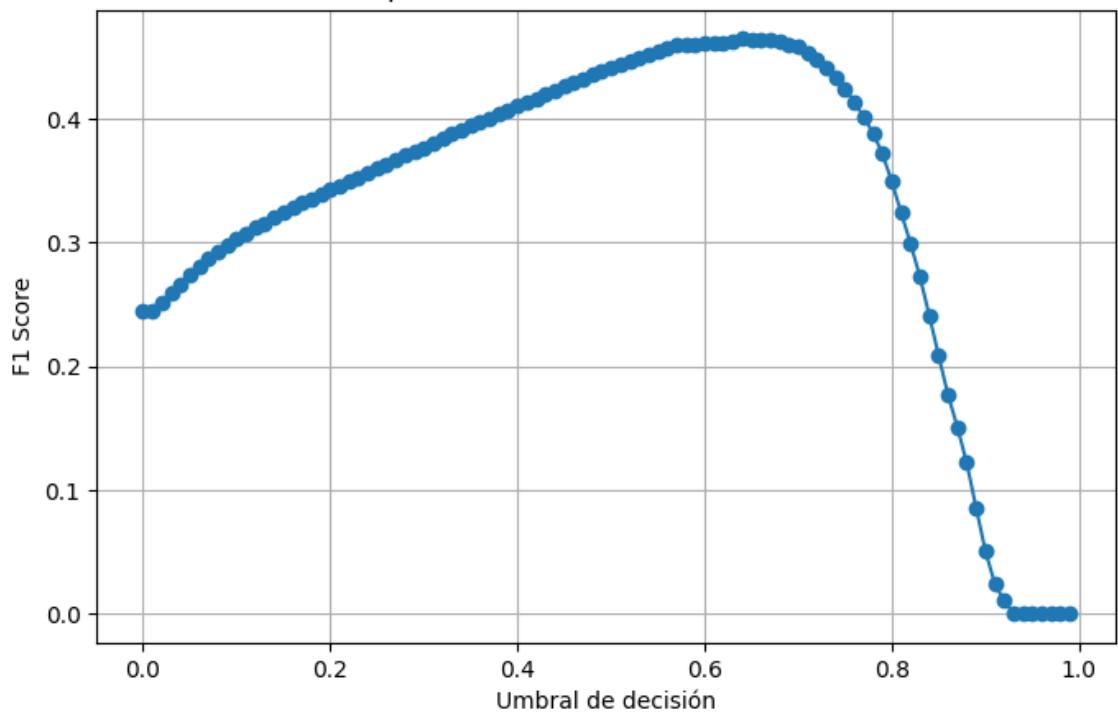
Por todo ello, se seleccionó como configuración óptima para el modelo final la Combinación 3, correspondiente a RandomizedSearchCV junto con is\_unbalance=True, ya que ofreció el mayor AUC (0.8260) y un equilibrio convincente entre rendimiento predictivo, estabilidad (score std = 0.0035) y capacidad para tratar eficazmente el desbalance de clases, a pesar de su mayor coste computacional (204.99 minutos).

### *Final model*

Una vez completadas las fases de ajuste de datos y de modelo, se construyó la versión definitiva del clasificador LightGBM utilizando la configuración óptima seleccionada previamente: todas las variables y balanceo de clases mediante el parámetro is\_unbalance=True. Esta combinación, correspondiente a la Combinación 3 de la Fase 2, fue seleccionada al alcanzar el mayor valor de AUC (0.8260) entre todas las variantes evaluadas, garantizando así el mejor rendimiento en la métrica principal de este estudio.

En esta fase final, no se repitió el proceso de optimización de hiperparámetros, ya que los valores óptimos se determinaron previamente y se aplicaron directamente. Se introdujo, sin embargo, una optimización del umbral de decisión con el objetivo de maximizar el F1-score. Como se observa en la figura correspondiente, el umbral óptimo se sitúa ligeramente por encima de 0.6, punto en el que se alcanza el mejor equilibrio entre precisión y sensibilidad, maximizando el F1.

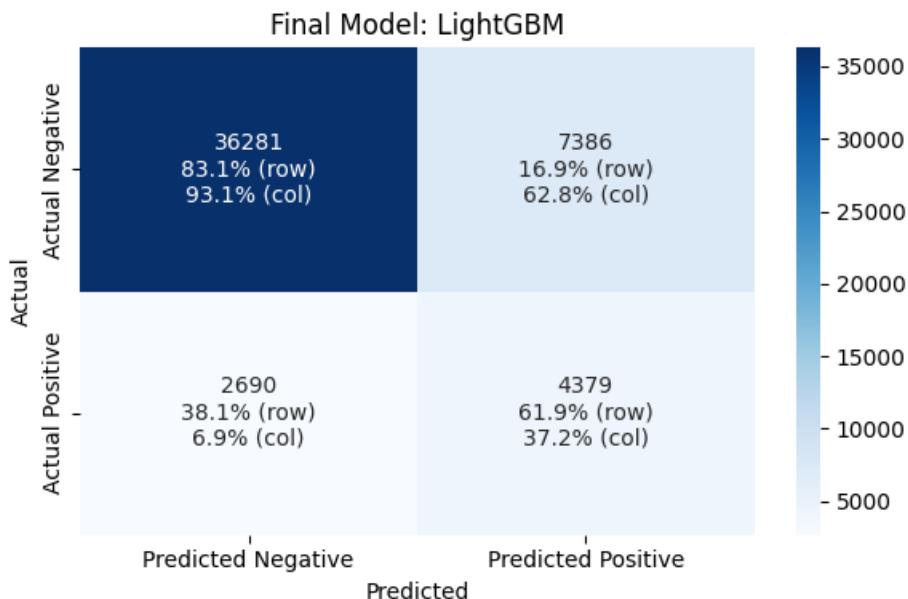
### Optimización del Umbral - F1 Score



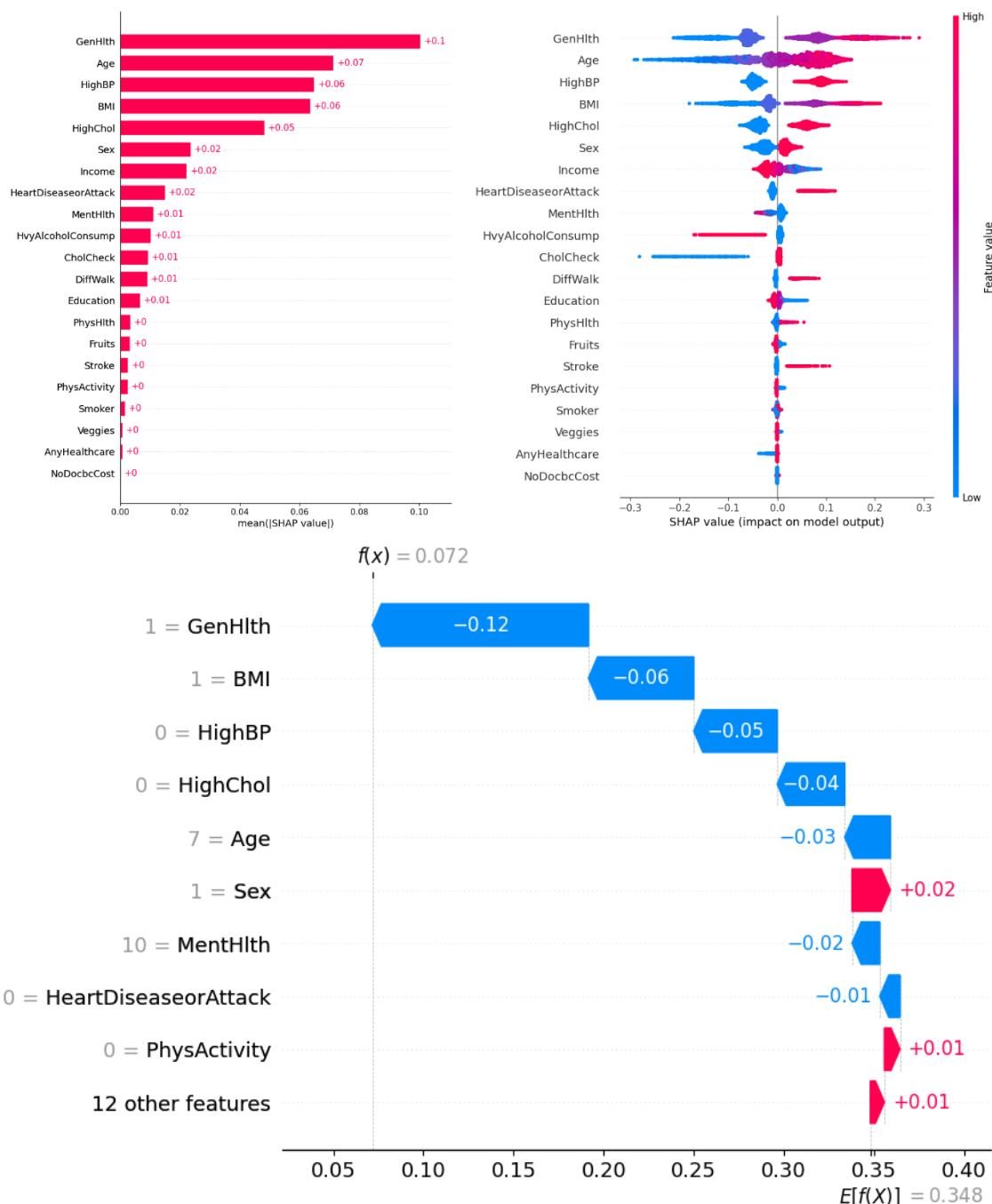
El rendimiento del modelo final se resume en la tabla de métricas: se alcanzó un AUC de 0.8260, una precisión global del 80.1%, un recall del 61.9% y una precisión de 37.2%, junto a un F1-score optimizado. Estos resultados indican que el modelo es capaz de identificar correctamente una proporción considerable de casos positivos, aunque a costa de un volumen moderado de falsos positivos. Esta relación se refleja claramente en la matriz de confusión, donde el modelo logra detectar 4.379 casos positivos correctamente, con 2.690 falsos negativos y 7.386 falsos positivos.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Final Model: LightGBM	0.826039	0.801403	0.619465	0.372206	0.829994	0.003525	0.346409



Desde la perspectiva de la interpretabilidad, se recurrió a técnicas basadas en SHAP para analizar las contribuciones de las variables al modelo. El gráfico de resumen por barras destaca a GenHlth, Age, HighBP, BMI y HighChol como los factores con mayor impacto en las predicciones. El gráfico de dispersión (summary dot plot) revela que valores elevados en estas variables (especialmente GenHlth y Age) aumentan significativamente la probabilidad de ser clasificado como prediabético o diabético. Finalmente, el gráfico tipo cascada (waterfall) ilustra detalladamente la predicción de un caso individual, mostrando cómo una combinación específica de valores llevó a una probabilidad elevada de clasificación positiva.



### III.6.f.- XGBoost

#### Fase 1

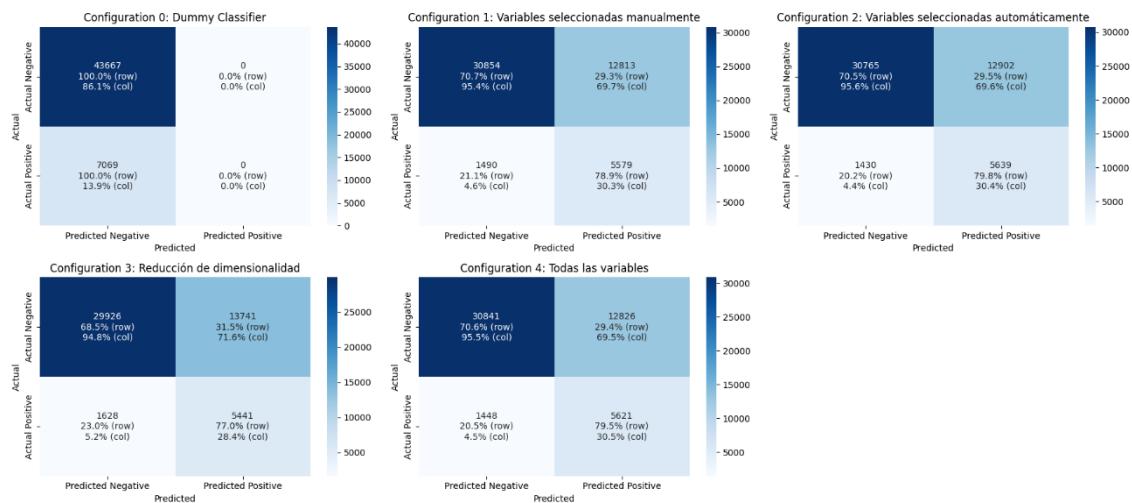
Durante la Fase 1 del ajuste del modelo LightGBM se analizaron diferentes estrategias de selección de variables con el objetivo de identificar la combinación que ofreciera el mejor rendimiento predictivo. En este análisis se evaluaron cuatro configuraciones basadas en distintos métodos de selección de atributos, además del modelo Dummy como referencia base.

Los mejores resultados en términos de área bajo la curva (AUC) se obtuvieron con la Configuración 4 (todas las variables), que alcanzó un valor de 0.825315, ligeramente por encima de la Configuración 2 (selección automática, AUC = 0.825117) y superior a la Configuración 1 (selección manual, AUC = 0.823104). Estas tres configuraciones superaron ampliamente al Dummy Classifier (AUC = 0.5). Por el contrario, la Configuración 3 (reducción de dimensionalidad) se quedó rezagada con un AUC de 0.7426, reflejando una pérdida de información relevante.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Configuration 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.001234
1	Configuration 1: Variables seleccionadas manualmente	0.823104	0.718090	0.789221	0.303338	0.827224	0.003296	5.573413
2	Configuration 2: Variables seleccionadas automáticamente	0.825117	0.717518	0.797708	0.304137	0.828946	0.003366	7.984365
3	Configuration 3: Reducción de dimensionalidad	0.748232	0.697079	0.769699	0.283651	0.755121	0.004914	302.614105
4	Configuration 4: Todas las variables	0.825315	0.718661	0.795162	0.304711	0.829339	0.003296	6.268311

Las matrices de confusión permiten observar con mayor detalle las diferencias entre configuraciones. Las configuraciones 1, 2 y 4 muestran un rendimiento muy similar en cuanto a recall ( $\approx 79.5\%$ ) y precisión ( $\approx 30.4\%$ ), destacando la Configuración 4 con un ligero mejor balance entre ambas métricas y un comportamiento más estable. En contraste, la Configuración 3 presenta un número más elevado de falsos positivos, reduciendo su precisión hasta el 28.4% y empeorando la exactitud total del modelo (69.7%).



Desde el punto de vista estadístico, los resultados obtenidos fueron evaluados con los tests de Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U. El test de Friedman mostró diferencias significativas entre las configuraciones ( $p < 0.000001$ ), y el test de Wilcoxon identificó diferencias relevantes entre la reducción de dimensionalidad y el resto de configuraciones ( $p$ -ajustado  $< 0.01$ ). Estos hallazgos fueron ratificados por los tests de Kruskal-Wallis y Mann-Whitney U, que confirman el bajo rendimiento de la Configuración 3 respecto al resto.

Resultados del Test de Friedman:  
 Estadístico Friedman p-value  
 39.28 6.097700e-08

Resultados del Test de Wilcoxon:  
 Primer Modelo Segundo Modelo Wilcox V p-value original p-value ajustado  
 Configuration 0: Dummy Classifier Configuration 1: man\_sel 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 2: aut\_sel 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 3: red\_dim 0.0 1.000000 1.000000  
 Configuration 0: Dummy Classifier Configuration 4: all 0.0 1.000000 1.000000  
 Configuration 1: man\_sel Configuration 2: aut\_sel 0.0 1.000000 1.000000  
 Configuration 1: man\_sel Configuration 3: red\_dim 55.0 0.000977 0.009766  
 Configuration 1: man\_sel Configuration 4: all 0.0 1.000000 1.000000  
 Configuration 2: aut\_sel Configuration 3: red\_dim 55.0 0.000977 0.009766  
 Configuration 2: aut\_sel Configuration 4: all 2.0 0.998047 1.000000  
 Configuration 3: red\_dim Configuration 4: all 0.0 1.000000 1.000000

Resultados del Test de Kruskal-Wallis:  
 Estadístico Kruskal-Wallis p-value  
 39.787241 4.789874e-08

Resultados del Test de Mann-Whitney U:  
 Primer Modelo Segundo Modelo U p-value original p-value ajustado  
 Configuration 0: Dummy Classifier Configuration 1: man\_sel 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 2: aut\_sel 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 3: red\_dim 0.0 0.999977 1.000000  
 Configuration 0: Dummy Classifier Configuration 4: all 0.0 0.999977 1.000000  
 Configuration 1: man\_sel Configuration 2: aut\_sel 27.0 0.962169 1.000000  
 Configuration 1: man\_sel Configuration 3: red\_dim 100.0 0.000091 0.000913  
 Configuration 1: man\_sel Configuration 4: all 23.0 0.981182 1.000000  
 Configuration 2: aut\_sel Configuration 3: red\_dim 100.0 0.000091 0.000913  
 Configuration 2: aut\_sel Configuration 4: all 44.0 0.688412 1.000000  
 Configuration 3: red\_dim Configuration 4: all 0.0 0.999933 1.000000

Por todo ello, se seleccionó como configuración óptima para la siguiente fase la Configuración 4, al presentar el mayor AUC (0.825315), mejor score medio (0.829339), baja desviación estándar (0.003296) y un tiempo de ejecución reducido (6.27 minutos), posicionándola como la opción más robusta y eficiente para el ajuste fino del modelo.

## Fase 2

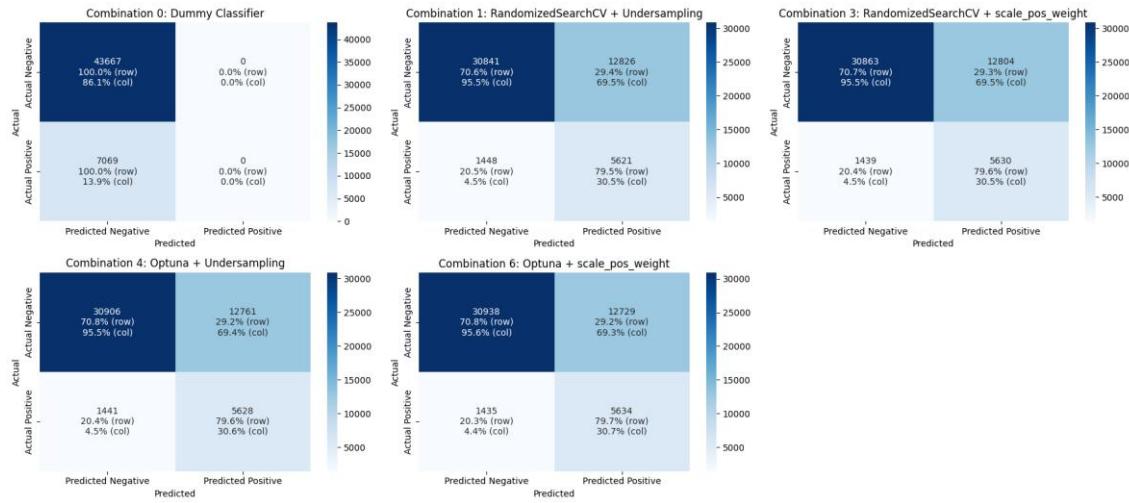
Durante la Fase 2 se exploraron distintas combinaciones de técnicas de balanceo y métodos de optimización de hiperparámetros para refinar el rendimiento del modelo. En total, se evaluaron cuatro combinaciones, que incluyeron RandomizedSearchCV y Optuna como optimizadores, junto con undersampling o el ajuste del parámetro scale\_pos\_weight.

Los resultados mostraron una alta estabilidad entre combinaciones, con AUCs comprendidos entre 0.825176 y 0.826165. La Combinación 3 (RandomizedSearchCV + scale\_pos\_weight) logró el mayor AUC (0.826165) y un score mean de 0.830058, lo que la posicionó como la variante más robusta. Además, mantuvo un equilibrio eficaz entre recall (79.6%) y precisión (30.5%), lo que sugiere un excelente rendimiento en entornos con desbalance de clases. La Combinación 6 (Optuna + scale\_pos\_weight) obtuvo métricas similares (AUC = 0.826030), pero con un coste computacional ligeramente superior al óptimo.

Summary Table:									
	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)	
0	Combination 0: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.000684	
1	Combination 1: RandomizedSearchCV + Undersampling	0.825315	0.718661	0.795162	0.304711	0.829339	0.003296	8.017184	
2	Combination 3: RandomizedSearchCV + scale_pos_weight	0.826165	0.719272	0.796435	0.305414	0.830058	0.003463	23.632361	
3	Combination 4: Optuna + Undersampling	0.825612	0.720080	0.796152	0.306053	0.829218	0.003470	6.046247	
4	Combination 6: Optuna + scale_pos_weight	0.826030	0.720829	0.797001	0.306813	0.830095	0.003557	21.289043	

Las matrices de confusión reflejan una sensibilidad consistente en todas las combinaciones útiles, con recall ≈ 79.7% y precisión ≈ 30.5%, valores adecuados considerando el sesgo de clases presente

en el dataset. La proporción entre verdaderos positivos y falsos positivos fue estable, destacando la confiabilidad del modelo en la detección de pacientes con riesgo de diabetes o prediabetes.



Desde el punto de vista estadístico, el test de Friedman detectó diferencias globales entre combinaciones ( $p < 0.000001$ ), mientras que los tests de Wilcoxon y Mann-Whitney U indicaron que las diferencias significativas se centraban en el contraste entre combinaciones con y sin técnicas de balanceo, en especial con respecto a la Combinación 4 (Optuna + undersampling), donde se observaron p-valores ajustados  $< 0.01$  frente a algunas variantes.

Resultados del Test de Friedman:

Estadístico Friedman	p-value
34.64	5.507380e-07

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcox V	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 4: Optuna + Un	0.0	1.000000	1.000000
Combination 0: Dummy Classifier	Combination 6: Optuna + wght	0.0	1.000000	1.000000
Combination 1: RandCV + Un	Combination 3: RandCV + wght	0.0	1.000000	1.000000
Combination 1: RandCV + Un	Combination 4: Optuna + Un	34.0	0.278320	1.000000
Combination 1: RandCV + Un	Combination 6: Optuna + wght	2.0	0.998047	1.000000
Combination 3: RandCV + wght	Combination 4: Optuna + Un	55.0	0.000977	0.009766
Combination 3: RandCV + wght	Combination 6: Optuna + wght	23.0	0.687500	1.000000
Combination 4: Optuna + Un	Combination 6: Optuna + wght	0.0	1.000000	1.000000

Resultados del Test de Kruskal-Wallis:

Estadístico Kruskal-Wallis	p-value
25.114753	0.000048

Resultados del Test de Mann-Whitney U:

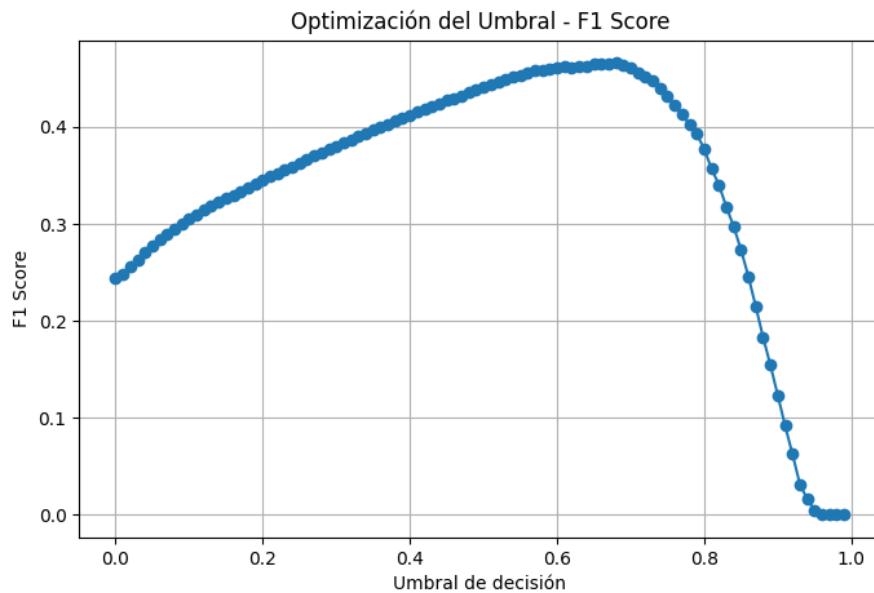
Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Combination 0: Dummy Classifier	Combination 1: RandCV + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 3: RandCV + wght	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 4: Optuna + Un	0.0	0.999977	1.0
Combination 0: Dummy Classifier	Combination 6: Optuna + wght	0.0	0.999977	1.0
Combination 1: RandCV + Un	Combination 3: RandCV + wght	40.0	0.786322	1.0
Combination 1: RandCV + Un	Combination 4: Optuna + Un	54.0	0.395668	1.0
Combination 1: RandCV + Un	Combination 6: Optuna + wght	37.0	0.846255	1.0
Combination 3: RandCV + wght	Combination 4: Optuna + Un	64.0	0.153745	1.0
Combination 3: RandCV + wght	Combination 6: Optuna + wght	49.0	0.545139	1.0
Combination 4: Optuna + Un	Combination 6: Optuna + wght	34.0	0.893853	1.0

En base a estos resultados, se seleccionó como configuración óptima la Combinación 3 (RandomizedSearchCV + scale\_pos\_weight) para la construcción del modelo final, por ofrecer el mejor AUC con una varianza baja ( $std = 0.003463$ ) y un tiempo de entrenamiento razonable (23.63 minutos).

### *Final model*

El modelo final XGBoost se entrenó utilizando la configuración completa de variables y la técnica de ajuste de clases mediante `scale_pos_weight`. Además, no se implementó Optuna ya que previamente se habían calculado y seleccionado los hiperparámetros óptimos en la fase anterior. En esta etapa final, se reutilizó directamente dicha configuración, omitiendo un nuevo proceso de ajuste.

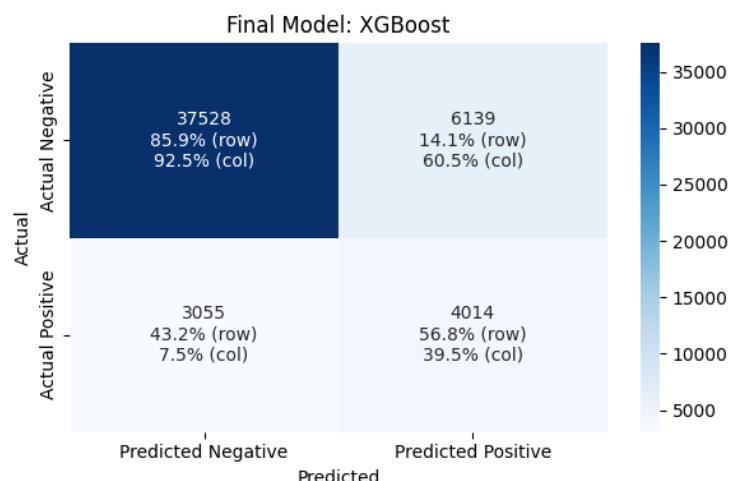
Se aplicó un procedimiento de **optimización del umbral de decisión** con el objetivo de maximizar el F1-score. El umbral óptimo resultó estar próximo a 0.68, valor a partir del cual se estabilizó el rendimiento del modelo y se observó un descenso abrupto en el score al incrementar el umbral.



El rendimiento alcanzado por el modelo fue notable, obteniéndose un valor de  $AUC = 0.826$ ,  $accuracy = 0.819$ ,  $recall = 0.568$ , y  $precision = 0.395$ . Estos resultados reflejan un buen compromiso entre sensibilidad y especificidad en el contexto del fuerte desbalance de clases del conjunto de datos. Además, el tiempo de ejecución fue muy reducido (0.086 minutos), lo que convierte a XGBoost en una opción especialmente eficiente en cuanto a coste computacional.

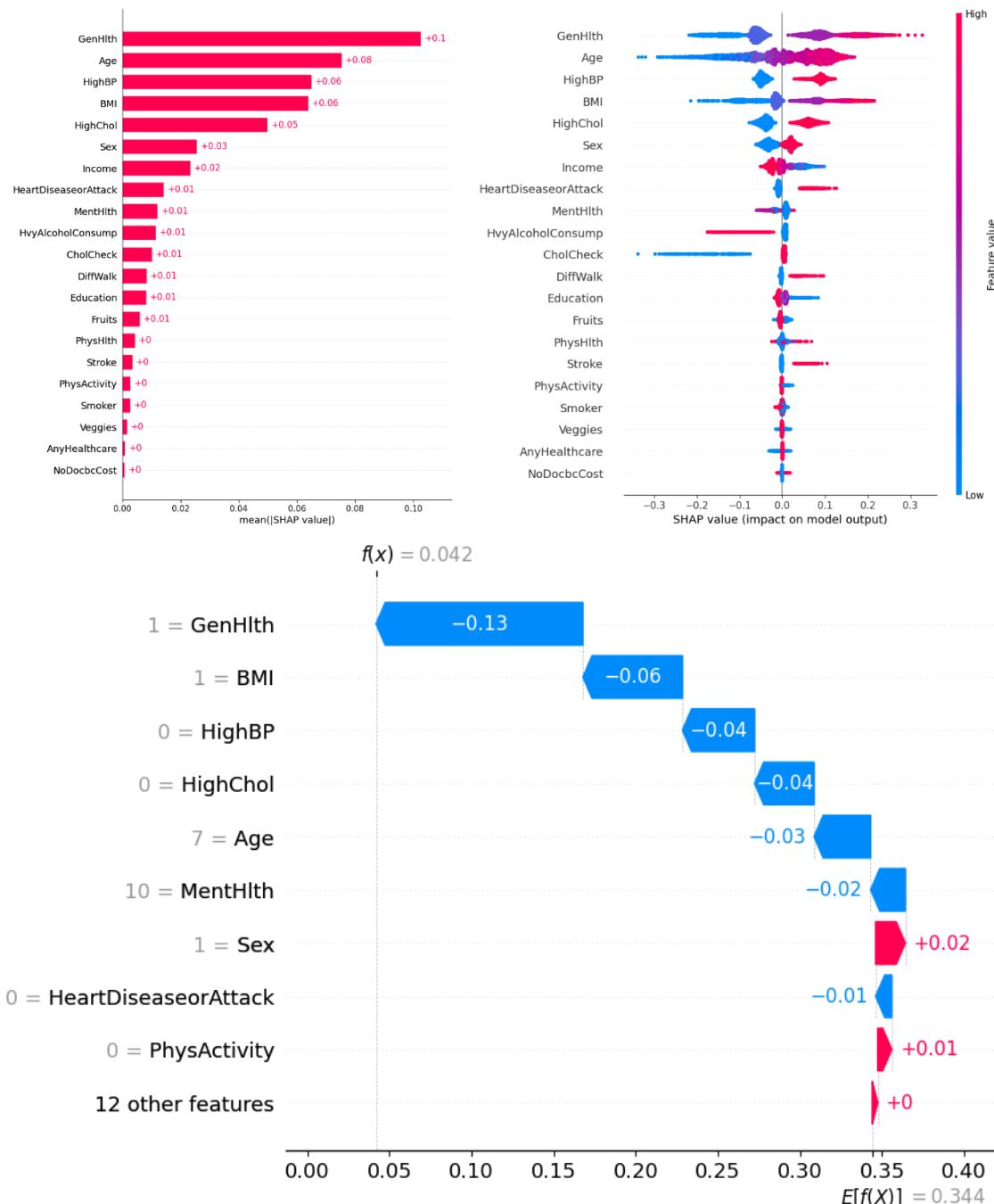
Summary Table:

Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
Final Model: XGBoost	0.826165	0.818787	0.567831	0.395351	0.830058	0.003463	0.08551



En cuanto a la interpretabilidad del modelo, el análisis de valores SHAP permitió identificar que las variables más influyentes fueron, en primer lugar, GenHlth, seguida de Age, HighBP, BMI y HighChol. En conjunto, estas cinco variables representaron el núcleo explicativo del modelo. Las gráficas de SHAP de tipo summary plot y dependence plot mostraron que los valores altos de estas variables se asocian típicamente con una mayor probabilidad de clasificación como diabético o prediabético.

Por último, se presentó un gráfico SHAP force plot para un caso individual, lo cual permitió analizar en detalle cómo cada variable contribuyó a la predicción concreta. En dicho ejemplo, el estado de salud general (GenHlth = 1) tuvo un efecto negativo decisivo en la predicción, disminuyendo fuertemente la probabilidad de clase positiva.

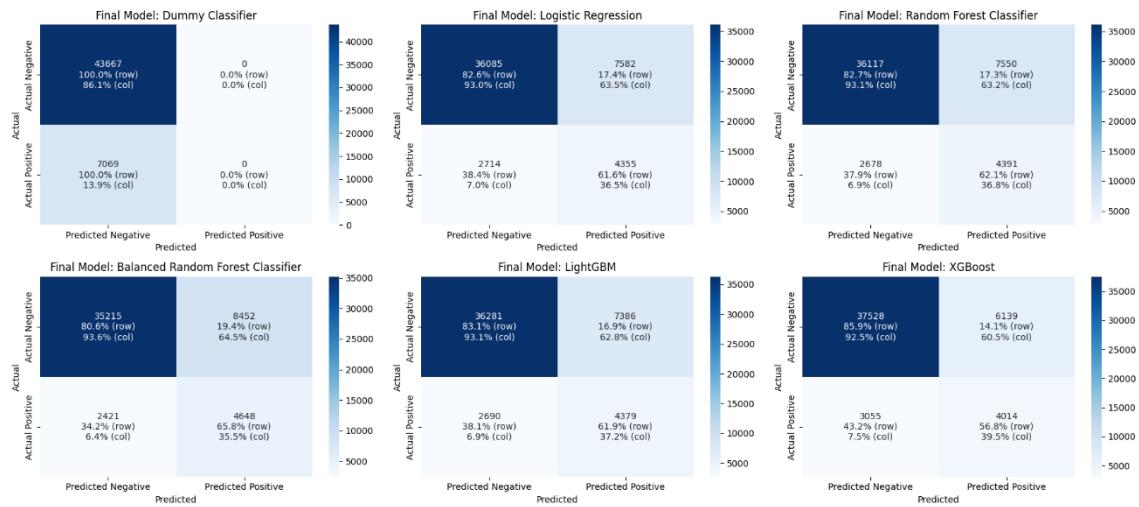


### III.6.g.- Resumen de modelos finales

En este apartado se presentan los resultados de todos los modelos finales alcanzados a lo largo de este proyecto.

Summary Table:

	Model	AUC	Accuracy	Recall	Precision	Score Mean	Score Std	Execution Time (minutes)
0	Final Model: Dummy Classifier	0.500000	0.860671	0.000000	0.000000	0.500000	0.000000	0.001570
1	Final Model: Logistic Regression	0.820267	0.797067	0.616070	0.364832	0.824262	0.002896	0.927780
2	Final Model: Random Forest Classifier	0.821267	0.798407	0.621163	0.367725	0.826084	0.003460	0.876639
3	Final Model: Balanced Random Forest Classifier	0.822478	0.785695	0.657519	0.354809	0.827014	0.003395	0.484018
4	Final Model: LightGBM	0.826039	0.801403	0.619465	0.372206	0.829994	0.003525	0.346409
5	Final Model: XGBoost	0.826165	0.818787	0.567831	0.395351	0.830058	0.003463	0.085510



Resultados del Test de Friedman:  
Estadístico Friedman p-value  
48.285714 3.105492e-09

Resultados del Test de Wilcoxon:

Primer Modelo	Segundo Modelo	Wilcoxon V	p-value original	p-value ajustado
Final Model 0: Dummy Classifier	Final Model 1: Logistic Regression	0.0	1.000000	1.0
Final Model 0: Dummy Classifier	Final Model 2: Random Forest Classifier	0.0	1.000000	1.0
Final Model 0: Dummy Classifier	Final Model 3: Balanced Random Forest Classifier	0.0	1.000000	1.0
Final Model 0: Dummy Classifier	Final Model 4: LightGBM	0.0	1.000000	1.0
Final Model 0: Dummy Classifier	Final Model 5: XGBoost	0.0	1.000000	1.0
Final Model 1: Logistic Regression	Final Model 2: Random Forest Classifier	2.0	0.998047	1.0
Final Model 1: Logistic Regression	Final Model 3: Balanced Random Forest Classifier	0.0	1.000000	1.0
Final Model 1: Logistic Regression	Final Model 4: LightGBM	0.0	1.000000	1.0
Final Model 1: Logistic Regression	Final Model 5: XGBoost	0.0	1.000000	1.0
Final Model 2: Random Forest Classifier	Final Model 3: Balanced Random Forest Classifier	0.0	1.000000	1.0
Final Model 2: Random Forest Classifier	Final Model 4: LightGBM	0.0	1.000000	1.0
Final Model 2: Random Forest Classifier	Final Model 5: XGBoost	0.0	1.000000	1.0
Final Model 3: Balanced Random Forest Classifier	Final Model 4: LightGBM	0.0	1.000000	1.0
Final Model 3: Balanced Random Forest Classifier	Final Model 5: XGBoost	0.0	1.000000	1.0
Final Model 4: LightGBM	Final Model 5: XGBoost	22.0	0.721680	1.0

Resultados del Test de Kruskal-Wallis:  
Estadístico Kruskal-Wallis p-value  
37.115076 5.679598e-07

Resultados del Test de Mann-Whitney U:

Primer Modelo	Segundo Modelo	U	p-value original	p-value ajustado
Final Model 0: Dummy Classifier	Final Model 1: Logistic Regression	0.0	0.999977	1.0
Final Model 0: Dummy Classifier	Final Model 2: Random Forest Classifier	0.0	0.999977	1.0
Final Model 0: Dummy Classifier	Final Model 3: Balanced Random Forest Classifier	0.0	0.999977	1.0
Final Model 0: Dummy Classifier	Final Model 4: LightGBM	0.0	0.999977	1.0
Final Model 0: Dummy Classifier	Final Model 5: XGBoost	0.0	0.999977	1.0
Final Model 1: Logistic Regression	Final Model 2: Random Forest Classifier	25.0	0.973049	1.0
Final Model 1: Logistic Regression	Final Model 3: Balanced Random Forest Classifier	21.0	0.987126	1.0
Final Model 1: Logistic Regression	Final Model 4: LightGBM	15.0	0.996358	1.0
Final Model 1: Logistic Regression	Final Model 5: XGBoost	15.0	0.996358	1.0
Final Model 2: Random Forest Classifier	Final Model 3: Balanced Random Forest Classifier	36.0	0.863482	1.0
Final Model 2: Random Forest Classifier	Final Model 4: LightGBM	17.0	0.994335	1.0
Final Model 2: Random Forest Classifier	Final Model 5: XGBoost	17.0	0.994335	1.0
Final Model 3: Balanced Random Forest Classifier	Final Model 4: LightGBM	20.0	0.989433	1.0
Final Model 3: Balanced Random Forest Classifier	Final Model 5: XGBoost	21.0	0.987126	1.0
Final Model 4: LightGBM	Final Model 5: XGBoost	47.0	0.604332	1.0

### III.7.- Resultado final y conclusiones

El presente trabajo ha permitido desarrollar un análisis integral de modelos predictivos orientados a la clasificación del estado de salud en relación con la diabetes y prediabetes a partir de un conjunto de datos poblacional. A lo largo de las distintas fases, se evaluaron enfoques basados en Random Forest, Balanced Random Forest, LightGBM y XGBoost, optimizando su rendimiento mediante técnicas de ajuste de hiperparámetros (RandomizedSearchCV y Optuna), estrategias de balanceo de clases (undersampling y class\_weight), selección de variables y ajuste del umbral de decisión. La validación del rendimiento de cada modelo fue rigurosamente respaldada mediante métricas de evaluación (AUC, precisión, recall, F1) y pruebas estadísticas no paramétricas (Friedman, Wilcoxon, Kruskal-Wallis y Mann-Whitney U).

Uno de los hallazgos más destacables del estudio es que, de manera sistemática y transversal en todos los modelos evaluados, la variable GenHlth —que refleja la percepción subjetiva del estado general de salud del paciente— emergió como la principal variable predictiva. Este resultado plantea una reflexión crítica sobre la viabilidad clínica y la robustez general de los modelos construidos. Si bien los modelos han mostrado una capacidad discriminativa sólida (valores de AUC en torno a 0.82 y scores F1 superiores a 0.82 en los mejores casos), su fiabilidad práctica podría verse comprometida si se utilizan en contextos donde la variable GenHlth no esté disponible o sea reportada de manera poco precisa.

La dependencia de una variable subjetiva como GenHlth puede introducir sesgos sistemáticos relacionados con la autoevaluación del estado de salud, los cuales varían considerablemente entre individuos debido a factores como nivel educativo, acceso a servicios sanitarios, cultura médica o incluso el estado emocional. En consecuencia, aunque los modelos son estadísticamente robustos, su trasladabilidad a la práctica clínica real requiere una cuidadosa revisión de las variables utilizadas y de la calidad con la que se registran.

Desde el punto de vista clínico, los modelos desarrollados podrían tener un valor sustancial como herramientas de cribado poblacional, especialmente en contextos de medicina preventiva o en estudios de cohortes amplias. Por ejemplo, podrían emplearse en plataformas de salud pública para identificar grupos de riesgo que requieren seguimiento médico adicional, intervenciones nutricionales o campañas de educación sanitaria. También serían útiles en aplicaciones móviles o sistemas de historia clínica electrónica como soporte a la toma de decisiones, siempre que las variables introducidas por el paciente estén debidamente validadas.

No obstante, la utilidad de estos modelos no depende exclusivamente de su rendimiento global, sino de cómo se priorizan sus métricas clave. En un entorno clínico, el recall (sensibilidad) se vuelve especialmente relevante: un modelo con alto recall es capaz de identificar con éxito la mayoría de los casos positivos, lo cual es crucial en enfermedades como la diabetes, donde el diagnóstico precoz puede reducir drásticamente complicaciones futuras. A su vez, mantener un nivel razonable de precisión evita un exceso de falsos positivos, lo cual es igualmente importante para no sobrecargar los sistemas de salud ni alarmar injustificadamente a la población.

Por esta razón, durante el ajuste del umbral de decisión se optó por maximizar el F1-score, que armoniza precisión y recall, buscando un equilibrio óptimo entre la sensibilidad diagnóstica y la fiabilidad de las alertas. Esta estrategia se traduce en una mayor aplicabilidad del modelo en contextos reales, donde los recursos son limitados y las decisiones deben ser justificables clínicamente.

En términos comparativos, el modelo que mejor equilibrio ofreció entre rendimiento predictivo, eficiencia computacional e interpretabilidad fue el modelo final XGBoost, con una AUC de 0.826, accuracy de 0.819, y un score F1 optimizado de 0.830. Además de ser el modelo más eficiente en términos de tiempo de ejecución, su capacidad explicativa, respaldada por el análisis de valores SHAP, permitió una interpretación coherente de las decisiones tomadas, lo cual constituye un aspecto clave para su aceptación en entornos clínicos.

En conclusión, este estudio demuestra que es viable construir modelos predictivos precisos y estables para la clasificación del estado de salud relacionado con la diabetes en grandes cohortes poblacionales. No obstante, se resalta la necesidad de revisar la naturaleza de las variables más influyentes, especialmente cuando estas son subjetivas, y de ajustar las métricas de evaluación en función del contexto clínico en el que se pretenda desplegar el modelo. La interpretación basada en SHAP y la validación estadística sólida fortalecen la fiabilidad de los resultados, pero también abren el camino a futuras investigaciones centradas en mejorar la recolección de datos objetivos, así como en integrar estos modelos en sistemas de soporte clínico reales.

## REFERENCIAS

---

OpenAI. (2025). *ChatGPT (versión GPT-4.0) [Large language model]*.

<https://www.openai.com/chatgpt>

*Scikit-learn: machine learning in Python — scikit-learn 1.6.1 documentation.* (s. f.). <https://scikit-learn.org/stable/>

*Imbalanced-learn documentation — Version 0.13.0.* (s. f.). <https://imbalanced-learn.org/stable/>

Google for Developers. (s. f.). *Google for Developers: Desde la IA y la nube hasta los dispositivos móviles y la Web.* Google For Developers. <https://developers.google.com/?hl=es>

*Welcome to LightGBM's documentation! — LightGBM 4.6.0 documentation.* (s. f.).

<https://lightgbm.readthedocs.io/en/stable/>

*XGBoost Documentation — xgboost 3.0.0 documentation.* (s. f.).

[https://xgboost.readthedocs.io/en/release\\_3.0.0/](https://xgboost.readthedocs.io/en/release_3.0.0/)

Content Studio. (2024, 29 mayo). *¿Qué son las métricas de rendimiento del aprendizaje automático?* Pure Storage. <https://www.purestorage.com/la/knowledge/machine-learning-performance-metrics.html>

*Métricas de evaluación.* (2025, 25 abril). 4Geeks. <https://4geeks.com/es/lesson/metricas-de-evaluacion>

*Ajuste del umbral según las características del proyecto - FasterCapital.* (s. f.). FasterCapital. <https://fastercapital.com/es/tema/ajuste-del-umbral-seg%C3%BAn-las-caracter%C3%ADsticas-del-proyecto.html>

*Welcome to the SHAP documentation — SHAP latest documentation.* (s. f.).

<https://shap.readthedocs.io/en/latest/>

Website, N. (2025, 26 marzo). *Obesity.* nhs.uk. <https://www.nhs.uk/conditions/obesity/>

Newest questions. (s. f.). Stack Overflow. <https://stackoverflow.com/questions>

Hinkle, D. E., Wiersma, W., & Jurs, S. G. (2003). Applied Statistics for the Behavioral Sciences. Boston, MA: Houghton Mifflin Company.

Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. Second Edition. Hillsdale, NJ: LEA.

Tomczak, E., Tomczak, M. (2014). The need to report effect size estimates revisited. An overview of some recommended measures of effect size. *TRENDS in Sport Sciences*, 21(1).

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.