

Práctica 0: Flujo completo de Machine Learning

Estimación de Niveles de Obesidad basado en Hábitos Alimenticios y Condición Física

Jordi Blasco Lozano

DNI: 74527208D

Universidad de Alicante - Escuela Politécnica Superior

Aprendizaje Avanzado - Curso 2025/2026

Email: jbl42@alu.ua.es

Resumen

En este trabajo se aborda un flujo completo de Machine Learning aplicado a la predicción de niveles de obesidad a partir de hábitos alimenticios y condición física. Se utiliza el dataset “Estimation of Obesity Levels” del repositorio UCI, compuesto por 2111 instancias y 17 variables (numéricas y categóricas). Se realiza un análisis exploratorio exhaustivo, un tratamiento de outliers en múltiples fases (IQR para Weight/Height, transformación Box-Cox para Age, y filtrado Z-score post-split), codificación de variables categóricas (Label Encoding y One-Hot Encoding) y normalización mediante Pipeline de scikit-learn con StandardScaler. Se entrena un modelo de Regresión Logística, se evalúa mediante validación cruzada 5-Fold y se optimiza con GridSearchCV, obteniendo un F1-Score final de 0.9359 en el conjunto de test.

1. Introducción

1.1. Dataset Seleccionado

Para esta práctica he seleccionado el dataset *Estimation of Obesity Levels Based on Eating Habits and Physical Condition* del repositorio UCI Machine Learning. Este dataset me pareció especialmente interesante por varias razones: primero, porque aborda un problema de salud pública muy relevante como es la obesidad; segundo, porque cumple a la perfección con todos los requisitos de la práctica; con variables numéricas y categóricas, con 7 categorías diferentes de peso lo que permite practicar diferentes técnicas de preprocesamiento; y tercero, porque el hecho de que la mayoría de los datos sean sintéticos (generados con SMOTE) añade un matiz interesante al análisis, ya que hay variables como NCP que presentan artefactos derivados de esa generación sintética.

El dataset está disponible en: <https://archive.ics.uci.edu/dataset/544>

Los datos fueron recopilados de individuos de México, Perú y Colombia mediante una plataforma web (23 % de los datos) y generados sintéticamente usando SMOTE en Weka (77 % restante). Este origen mixto real-sintético tiene implicaciones directas en el preprocesamiento, como veremos en el tratamiento de la variable NCP.

1.2. Objetivo

El objetivo principal de este trabajo es predecir el nivel de obesidad de una persona basándose en sus características demográficas, hábitos alimenticios y nivel de actividad física. Se trata de un problema de clasificación multiclasa con 7 categorías, lo cual requiere métricas adecuadas y técnicas de codificación apropiadas. Las clases a predecir son:

- Peso Insuficiente (*Insufficient_Weight*)
- Peso Normal (*Normal_Weight*)

- Sobre peso Nivel I (*Overweight_Level_I*)
- Sobre peso Nivel II (*Overweight_Level_II*)
- Obesidad Tipo I (*Obesity_Type_I*)
- Obesidad Tipo II (*Obesity_Type_II*)
- Obesidad Tipo III (*Obesity_Type_III*)

El flujo de trabajo sigue la estructura propuesta del enunciado: exploración, preprocesamiento, entrenamiento con Pipeline, validación cruzada y optimización de hiperparámetros.

2. Configuración Experimental

2.1. Exploración Inicial

Lo primero que hice fue cargar el dataset y examinar sus características básicas. Las propiedades del conjunto de datos se resumen en la Tabla 1.

Tabla 1: Características del dataset de Obesidad

Característica	Valor
Filas	2111
Columnas	17 (16 features + 1 target)
Var. numéricas	8 (Age, Height, Weight, FCVC, NCP, CH2O, FAF, TUE)
Var. categóricas	8 (Gender, family_history, FAVC, CAEC, SMOKE, SCC, CALC, MTRANS)
Variable objetivo	NObeyesdad
Tipo problema	Clasificación multiclas (7 clases)
Valores faltantes	0

Es importante destacar que el dataset no tiene ningún valor faltante, lo cual es poco frecuente en datasets reales y se debe a que la mayoría de los datos fueron generados sintéticamente. Esto nos permite centrar el preprocesamiento en el tratamiento de outliers y en la codificación de variables, sin necesidad de aplicar estrategias de imputación.

2.2. Análisis Exploratorio (EDA)

2.2.1. Estadística Descriptiva

Calculé las estadísticas descriptivas de las variables numéricas principales. La Tabla 2 muestra un resumen de las más relevantes. Lo que más llama la atención es el rango de la variable Weight (39–173 kg), que tiene todo el sentido dado que estamos clasificando desde peso insuficiente hasta obesidad tipo III. También es interesante que la media de Age es de 24.31 años con una desviación estándar de 6.35, lo que indica una población relativamente joven y con una distribución sesgada a la derecha. Se encuentran personas de hasta 61 años pero la mayoría rondan los 20–25.

Tabla 2: Estadísticas descriptivas de variables numéricas principales

Variable	Media	Std	Min	Mediana	Max
Age	24.31	6.35	14.0	22.8	61.0
Height	1.70	0.09	1.45	1.70	1.98
Weight	86.59	26.19	39.0	83.0	173.0
FCVC	2.42	0.53	1.0	2.39	3.0
NCP	2.69	0.78	1.0	3.0	4.0
CH2O	2.01	0.61	1.0	2.0	3.0
FAF (Actividad física)	1.01	0.85	0.0	1.0	3.0
TUE	0.66	0.61	0.0	0.63	2.0

En cuanto a las variables categóricas, observé que el dataset presenta una distribución de género bastante equilibrada (1068 hombres vs 1043 mujeres). Un dato que me llamó la atención es que el 81.8 % de los individuos tiene antecedentes familiares de sobrepeso (`family_history_with_overweight = yes`), lo cual tiene sentido en un dataset enfocado en obesidad. Además, el 88.4 % consume frecuentemente alimentos calóricos (`FAVC = yes`) y el transporte público es el medio más utilizado con un 74.8 %.

2.2.2. Distribuciones

Generé histogramas de todas las variables numéricas para entender sus distribuciones (Figura 1). Lo que observé es que la variable Weight presenta una distribución bastante amplia y relativamente uniforme, lo cual tiene sentido dado que estamos clasificando desde peso insuficiente hasta obesidad tipo III: al haber 7 categorías de obesidad, es natural que la distribución cubra un rango muy amplio.

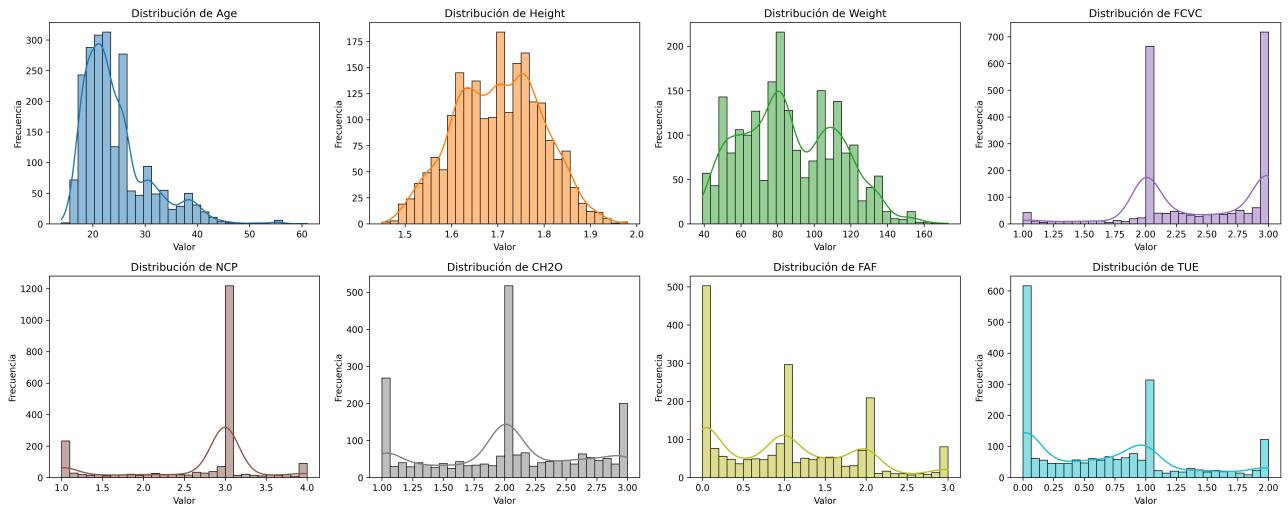


Figura 1: Distribuciones de las variables numéricas del dataset. Se observa que Age presenta un sesgo positivo (cola a la derecha), Height sigue una distribución relativamente normal, y Weight cubre un rango amplio acorde a las 7 clases.

Las variables Age y Height siguen distribuciones relativamente normales, aunque Age presenta un claro sesgo a la derecha (muchas personas jóvenes y pocas mayores de 40). Las variables FAF (frecuencia de actividad física) y TUE (tiempo de uso de dispositivos tecnológicos) presentan distribuciones más sesgadas, con la mayoría de valores concentrados en la parte baja del rango. Un detalle interesante es que las variables FCVC, NCP y CH2O muestran picos en valores enteros, lo cual es un artefacto del proceso de generación sintética con SMOTE: los datos reales de encuestas son discretos (1, 2, 3), mientras que SMOTE genera interpolaciones decimales entre ellos.

2.2.3. Correlaciones

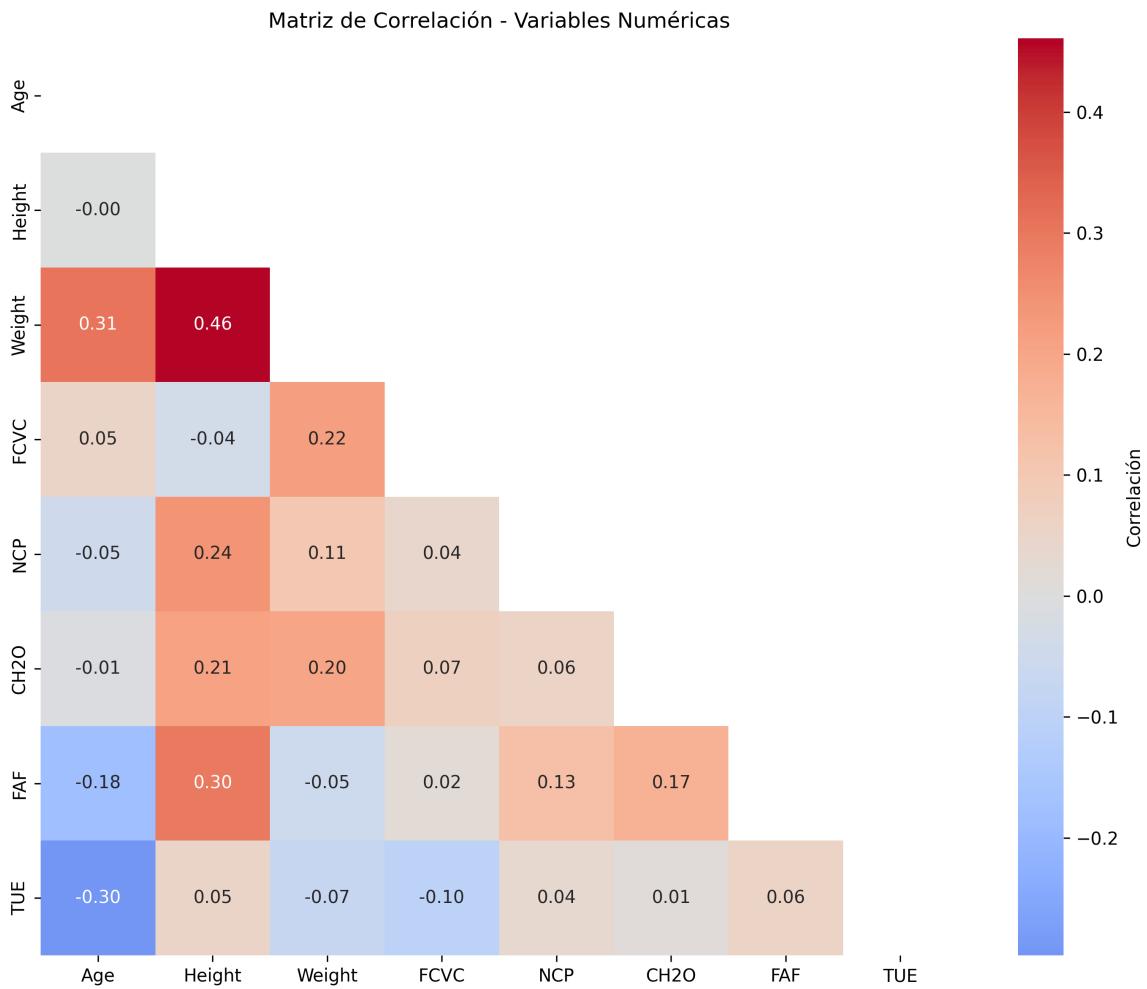


Figura 2: Matriz de correlación entre variables numéricas. Se destaca la correlación moderada-fuerte entre Weight y Height (0.46), y correlaciones moderadas de Weight con Age (0.31) y FAF con Height (0.30).

El análisis de correlaciones (Figura 2) reveló varios patrones interesantes. La correlación más fuerte es entre **Weight y Height** ($r = 0,46$), lo cual es esperable desde el punto de vista fisiológico: personas más altas tienden a pesar más. También encontré una correlación moderada entre **Weight y Age** ($r = 0,31$), sugiriendo que el peso tiende a aumentar con la edad. Me llamó la atención la correlación **negativa entre Age y TUE** ($r = -0,30$), indicando que las personas más jóvenes dedican más tiempo a dispositivos tecnológicos, algo muy coherente con la realidad actual.

Otra correlación interesante es la de **FAF (actividad física) y Height** ($r = 0,30$), que podría explicarse porque las personas más altas tienden a participar más en actividades deportivas. Por otro lado, las correlaciones entre variables de hábitos alimenticios (FCVC, NCP, CH2O) son generalmente débiles entre sí ($r < 0,10$), lo que sugiere que cada hábito aporta información independiente al modelo.

Para complementar la matriz de correlación, generé scatter plots de las combinaciones más relevantes coloreados por clase de obesidad (Figura 3).

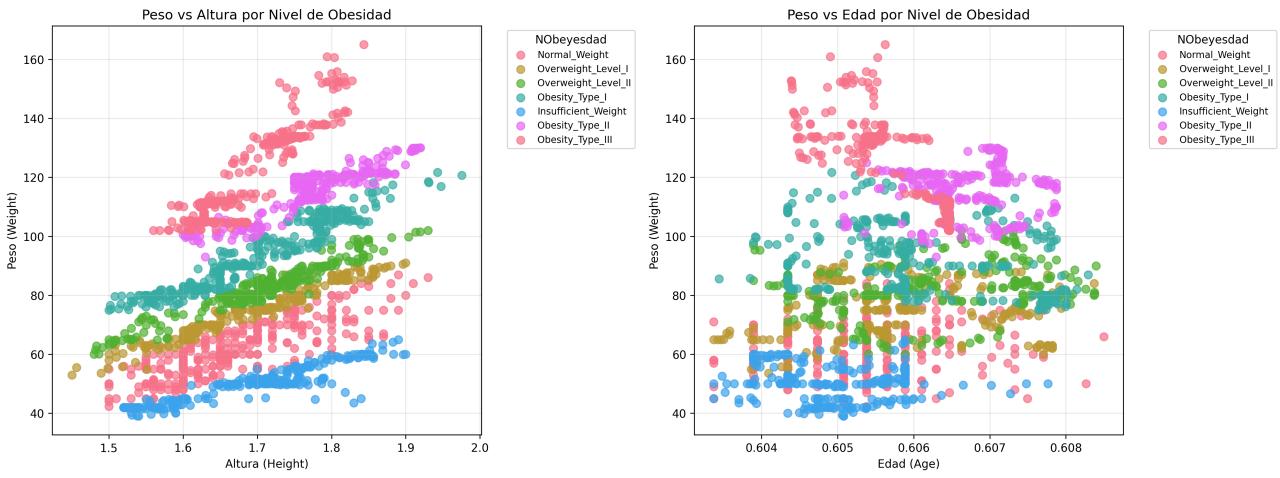


Figura 3: Scatter plots de Peso vs Altura y Peso vs Edad, coloreados por nivel de obesidad. Se observa una clara separación de clases en el plano Peso-Altura (izquierda), mientras que en Peso-Edad (derecha) las clases se superponen considerablemente.

El scatter plot de **Peso vs Altura** es especialmente revelador: las categorías de obesidad se organizan en bandas diagonales ascendentes, con una separación bastante clara entre ellas. Esto tiene todo el sentido físico, ya que el índice de masa corporal (IMC) se calcula directamente como $IMC = Peso/Altura^2$, por lo que estas dos variables son los predictores más potentes del modelo. En cambio, el scatter plot de **Peso vs Edad** muestra que la edad por sí sola no es un factor determinante para clasificar el nivel de obesidad: las clases están mucho más mezcladas, indicando que la edad debe combinarse con otras variables para ser informativa.

2.2.4. Balance de Clases

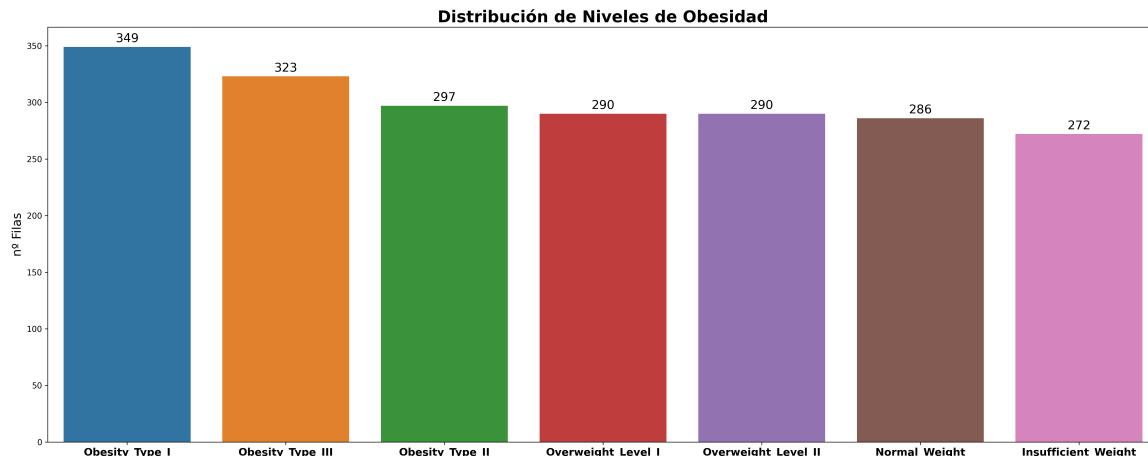


Figura 4: Distribución de los niveles de obesidad en el dataset tras el tratamiento de outliers. Las clases están razonablemente balanceadas, oscilando entre 272 y 349 muestras.

Observé que las clases están razonablemente balanceadas (Figura 4), con cada una representando entre el 12.9 % y el 16.6 % del total de muestras. La clase más representada es Obesity_Type_I con 349 instancias, y la menos representada es Insufficient_Weight con 272. Esta distribución relativamente equilibrada es favorable para el entrenamiento de los modelos, ya que no necesitamos aplicar técnicas de balanceo como SMOTE o undersampling. Si las clases estuviesen muy desbalanceadas, el modelo tendería a predecir siempre la clase mayoritaria, inflando artificialmente el accuracy pero fallando en las clases minoritarias.

2.2.5. Visualización PCA

Para obtener una visión global de la separabilidad de las clases, apliqué un Análisis de Componentes Principales (PCA) reduciendo a 2 dimensiones las 8 variables numéricas estandarizadas. El resultado se muestra en la Figura 5.

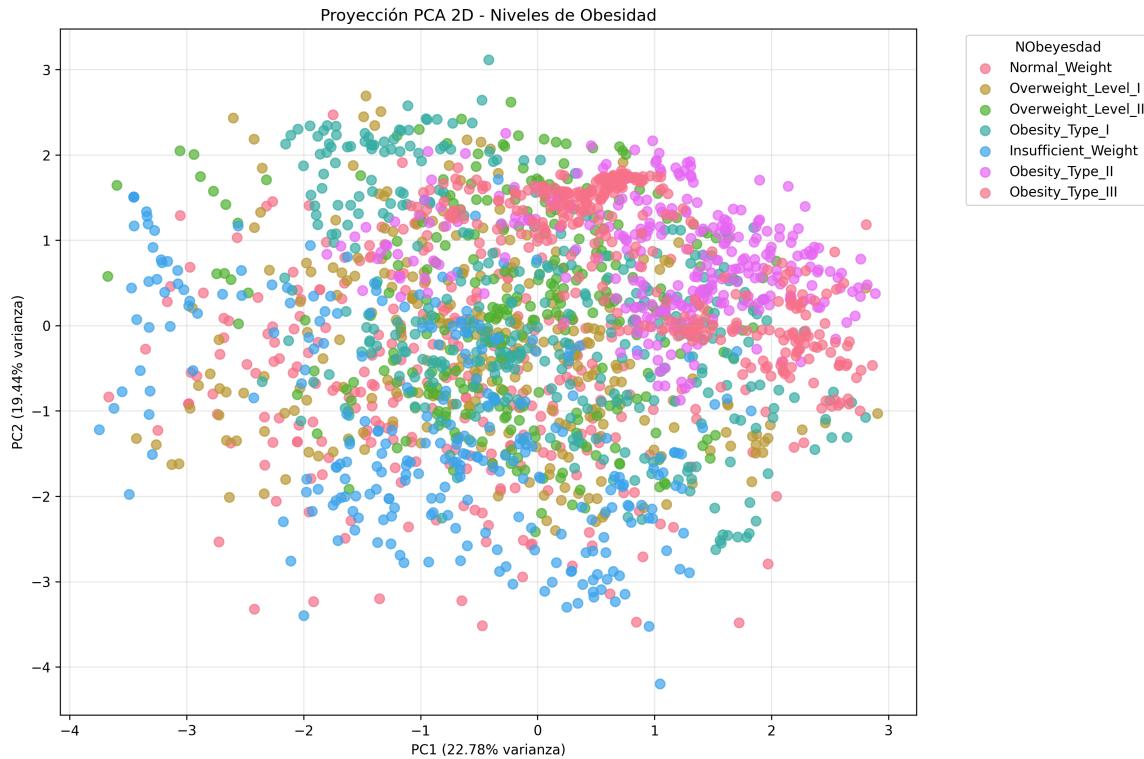


Figura 5: Proyección PCA 2D de las variables numéricas, coloreada por nivel de obesidad. Las dos primeras componentes capturan una fracción limitada de la varianza total, y las clases se superponen significativamente.

Lo que el PCA nos muestra es que los datos claramente no son linealmente separables en esta proyección 2D. Las distintas clases (nubes de puntos) se superponen significativamente unas con otras, y no es posible trazar una línea recta que separe una clase del resto sin cometer numerosos errores. Esto tiene dos implicaciones importantes: primera, que un clasificador lineal como la Regresión Logística tendrá limitaciones inherentes (aunque puede compensarlas parcialmente con las variables categóricas que no aparecen en este PCA); y segunda, que las dos primeras componentes principales capturan una fracción limitada de la varianza total, lo que significa que la información discriminante está distribuida a lo largo de múltiples dimensiones.

2.2.6. Hallazgos del EDA

A modo de resumen, los principales hallazgos del análisis exploratorio son:

1. **Weight y Height son los predictores más potentes:** Su combinación permite separar visualmente las clases de obesidad en bandas diagonales, coherente con la definición del IMC.
2. **Age presenta un sesgo positivo notable:** La mayoría de individuos tiene entre 18 y 30 años, con una cola larga hacia edades mayores. Esto justificará la transformación Box-Cox que aplicaremos.
3. **NCP presenta artefactos de SMOTE:** Los picos en valores enteros (datos reales) mezclados con valores decimales (datos sintéticos) crean una distribución multimodal artificial.

4. **Las clases están razonablemente balanceadas:** Ninguna clase domina sobre las demás, lo que simplifica el entrenamiento.
5. **Los datos no son linealmente separables en 2D:** El PCA confirma superposición entre clases, sugiriendo que el modelo necesita todas las dimensiones (incluyendo categóricas) para discriminar eficazmente.
6. **Las correlaciones entre variables de hábitos son débiles:** Cada variable aporta información diferente, lo que justifica mantener todas en el modelo.

2.3. Preprocesamiento

2.3.1. Valores Faltantes

El dataset no presenta valores faltantes, por lo que no fue necesario aplicar ninguna estrategia de imputación.

2.3.2. Outliers

El tratamiento de outliers lo dividí en **tres fases**, cada una con una justificación diferente. La detección inicial la realicé con el método IQR (Rango Intercuartílico), que es robusto frente a distribuciones sesgadas. Los resultados se resumen en la Tabla 3.

Tabla 3: Outliers detectados por variable (método IQR)

Variable	Outliers	%
Age	168	7.96 %
Height	1	0.05 %
Weight	1	0.05 %
FCVC	0	0.00 %
NCP	579	27.43 %
CH2O	0	0.00 %
FAF	0	0.00 %
TUE	0	0.00 %

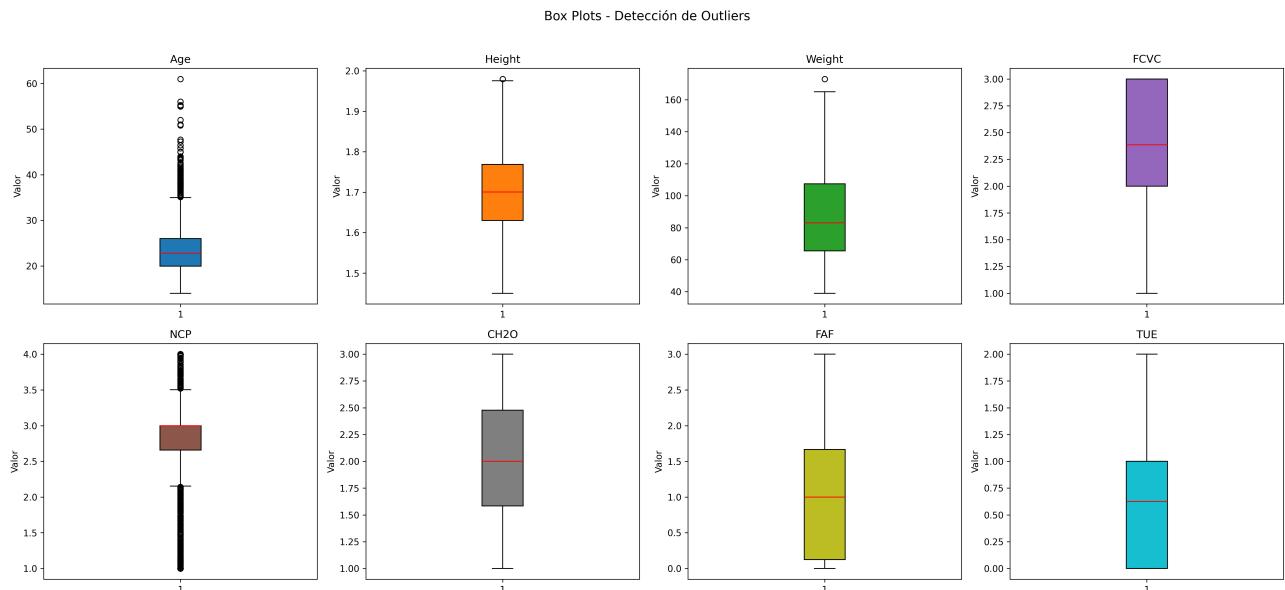


Figura 6: Box plots de las variables numéricas para detección visual de outliers. Se observan claramente los outliers de Age (cola superior) y la distribución peculiar de NCP con valores extremos en el IQR.

Fase 1: Eliminación de outliers de Weight y Height. Para estas dos variables, los outliers detectados por IQR fueron mínimos (1 valor atípico en cada una). Decidí eliminar directamente las filas correspondientes, ya que representan casos extremos aislados que podrían distorsionar la estandarización posterior. El resultado fue la eliminación de 2 filas, un impacto negligible en el tamaño del dataset.

Fase 2: Transformación Box-Cox para Age. La variable Age presentaba 168 outliers (7.96 %), todos concentrados en la cola derecha de la distribución (personas mayores de ~ 35 años). En lugar de eliminar directamente estas 168 filas (lo que supondría perder un 8 % de los datos), opté por aplicar una transformación Box-Cox para corregir el sesgo de la distribución antes de filtrar outliers.

Box-Cox es una familia de transformaciones de potencia parametrizadas por un valor λ que busca convertir una distribución no normal en una distribución lo más cercana posible a la normal:

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{si } \lambda \neq 0 \\ \ln(y) & \text{si } \lambda = 0 \end{cases} \quad (1)$$

El λ se obtiene mediante **máxima verosimilitud** (Maximum Likelihood Estimation). La idea es la siguiente: se prueban muchos valores de λ (por ejemplo, de -5 a 5) y para cada uno se aplica la transformación a los datos, se calcula cuánto se parece la distribución resultante a una distribución normal (usando la función de log-verosimilitud), y se elige el λ que maximiza esa similitud. En la práctica, la función `scipy.stats.boxcox()` de Python hace todo este proceso automáticamente: internamente optimiza λ para que los datos transformados sean lo más normales posible, y nos devuelve tanto los datos transformados como el λ óptimo encontrado.

El λ óptimo encontrado fue $\lambda \approx -1.64$, lo que implica una corrección muy fuerte: más agresiva que un logaritmo ($\lambda = 0$), más cercana a una inversa potenciada (proporcional a $1/x^{1.64}$). El signo negativo indica una relación inversa, y la magnitud ($| -1.64 | > 1$) refleja la intensidad de la compresión necesaria para normalizar la cola derecha. La ventaja de usar Box-Cox frente a elegir manualmente logaritmo o raíz cuadrada es que Box-Cox encuentra el exponente exacto que maximiza la normalidad, minimizando así el número de filas que terminamos eliminando como outliers.

La Figura 7 muestra el efecto de la transformación sobre la distribución de Age.

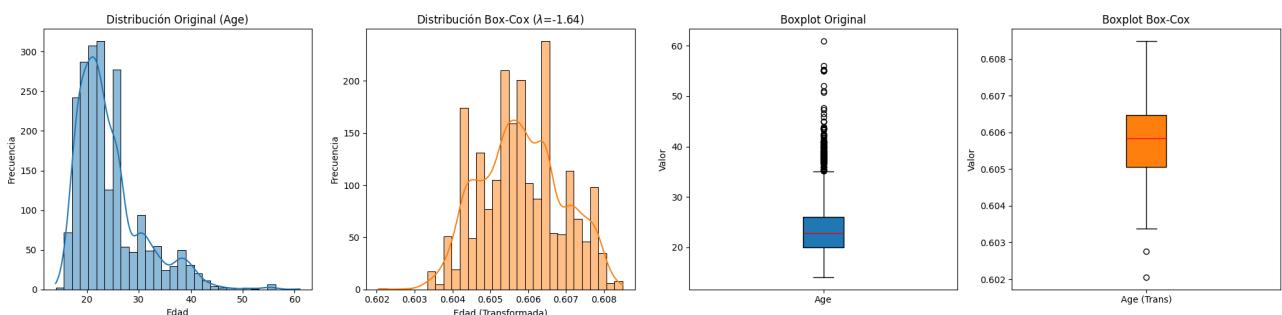


Figura 7: Comparación antes y después de la transformación Box-Cox sobre la variable Age. A la izquierda se muestran el histograma y boxplot originales, con un claro sesgo positivo y numerosos outliers en la cola derecha. A la derecha, tras aplicar Box-Cox con $\lambda = -1.64$, la distribución se comprime y simetriza notablemente, reduciendo los outliers detectados por IQR de 168 a solo 2.

En el panel izquierdo (distribución original) se aprecia claramente el sesgo positivo de Age: la masa de datos se concentra en el rango 15–30 años, con una cola larga hacia la derecha que llega hasta los 61 años. El boxplot original confirma esta asimetría, mostrando una gran cantidad de puntos marcados como outliers por encima de la línea superior. En el panel derecho (distribución transformada), Box-Cox ha comprimido esa cola derecha de forma muy agresiva, concentrando los valores en un rango estrecho (~ 0.602 – 0.608). El boxplot transformado muestra que ahora solo quedan unos pocos outliers aislados en la parte inferior, que son los que finalmente eliminamos.

Tras la transformación, recalcué los outliers por IQR sobre la distribución transformada y solo fue necesario eliminar 2 filas adicionales, frente a las 168 que habría eliminado sin transformar. Esto supone una reducción del 98.8% en la pérdida de datos.

Fase 3: Decisión sobre NCP. La variable NCP (Number of Main Meals) presentaba 579 outliers detectados por IQR (27.43 %), una cifra alarmante. Sin embargo, tras analizar la distribución en detalle, decidí no aplicar ningún tratamiento. La razón es que NCP es conceptualmente una variable discreta (número de comidas: 1, 2, 3 o 4), y los “outliers” detectados son en realidad artefactos del algoritmo SMOTE: los datos reales (enteros) generan picos en 1, 3 y 4, mientras que SMOTE genera interpolaciones decimales entre estos valores. Aplicar transformaciones suaves (logaritmo, raíz cuadrada) no consigue normalizar una distribución con esta estructura multimodal artificial, por lo que opté por conservar la variable original y confiar en la estandarización del Pipeline para manejar las escalas.

2.3.3. Transformaciones y Codificación

Una vez tratados los outliers, el siguiente paso fue preparar las variables para los algoritmos de Machine Learning. Los modelos numéricos no entienden texto (“Male”, “Sometimes”), por lo que es necesario transformar las variables categóricas a valores numéricos. Aplicué dos estrategias diferentes según la naturaleza de cada variable:

- **Label Encoding** para variables binarias: Gender, family_history_with_overweight, FAVC, SMOKE y SCC. Al tener solo dos categorías posibles, el Label Encoding (asignar 0 y 1) no introduce ningún orden falso y es la opción más eficiente en términos de dimensionalidad.
- **One-Hot Encoding** para variables categóricas multinivel: CAEC (consumo de alimentos entre comidas), CALC (consumo de alcohol) y MTRANS (medio de transporte). Estas variables tienen 3–5 categorías sin orden jerárquico natural. Si usásemos Label Encoding (ej: Walking=0, Bike=1, Public_Transportation=2), el modelo interpretaría que “Public_Transportation > Bike > Walking”, lo cual es un orden falso que sesgaría el aprendizaje. One-Hot Encoding evita este problema creando columnas binarias independientes para cada categoría. Usé `drop_first=True` para evitar multicolinealidad perfecta (la primera categoría queda implícita cuando todas las demás son 0).

Tras la codificación, el dataset pasó de 16 features originales a 23 features (las 8 numéricas + 5 binarias codificadas + 10 columnas de One-Hot Encoding). La variable objetivo (NObeyesdad) se codificó con Label Encoding, asignando valores de 0 a 6 para las 7 clases.

2.3.4. Estandarización

Es importante destacar que la estandarización (Z-score) no se aplicó manualmente en esta fase, sino que se gestiona internamente dentro del Pipeline de scikit-learn (ver Sección 2.5).

Al encadenar `StandardScaler` + modelo dentro del Pipeline, se garantiza que: (1) al llamar a `pipeline.fit(X_train, y_train)`, el scaler aprende μ y σ solo de los datos de entrenamiento; (2) al llamar a `pipeline.predict(X_test)`, se aplica la misma transformación al test usando las estadísticas del train. Esto elimina por completo el riesgo de Data Leakage, ya que es imposible que las estadísticas del test contaminen el entrenamiento.

2.4. Train-Test Split

Dividí los datos en 80% para entrenamiento y 20% para test, utilizando estratificación para mantener la proporción de clases en ambos subconjuntos. Esto es fundamental: si no estratificamos, podríamos tener un test con muy pocas muestras de alguna clase (por ejemplo, solo 2 muestras de Insufficient_Weight), lo que haría que la evaluación de esa clase fuese poco fiable.

Tabla 4: División de datos

Conjunto	N	%
Train	1685	80
Test	422	20

Tras la división, verifiqué que la distribución de clases en el conjunto de entrenamiento se mantuviese proporcional al dataset completo. Los porcentajes por clase oscilan entre el 12.9% (Clase 0: Insufficient_Weight) y el 16.6% (Clase 2: Obesity_Type_I), confirmando que la estratificación funciona correctamente.

Además, tras el split apliqué un filtrado de outliers adicional por Z-score únicamente sobre el conjunto de entrenamiento. La idea es calcular los Z-scores de las variables numéricas de forma temporal (sin guardar la transformación) y eliminar las muestras cuyo Z-score excede ± 3 en cualquier variable. En este caso particular, el filtrado no eliminó ninguna muestra adicional (0 filas eliminadas), lo que indica que los tratamientos previos de outliers (fases 1–3) fueron suficientes para limpiar los casos extremos.

2.5. Pipeline

Uno de los aspectos más importantes de este trabajo es el uso de Pipelines de scikit-learn, que es la forma profesional y segura de encadenar transformaciones y modelos. Un Pipeline encadena múltiples pasos en un único objeto, de forma que al llamar a `.fit()` se ejecutan todos los pasos secuencialmente, y al llamar a `.predict()` se aplican las transformaciones aprendidas antes de predecir.

En nuestro caso, el Pipeline tiene dos pasos:

1. **StandardScaler**: Estandariza las features (media = 0, std = 1).
2. **LogisticRegression**: El modelo de clasificación.

¿Por qué usar Pipeline en lugar de hacer el escalado manualmente? La razón principal es que previene el Data Leakage de forma automática. Sin Pipeline, es muy fácil cometer el error de hacer `scaler.fit_transform(X_completo)` antes de dividir en train/test, contaminando el test con estadísticas del train. Con Pipeline, esto es imposible: el scaler solo ve los datos de entrenamiento durante `.fit()`, y cuando usamos `cross_validate` o `GridSearchCV`, en cada fold la estandarización se recalcula de forma independiente.

Además, el Pipeline facilita enormemente la optimización de hiperparámetros: podemos pasar el Pipeline directamente a `GridSearchCV` y tunear parámetros del modelo usando la notación `model__C`, `model__max_iter`, etc.

2.6. Modelo Evaluado

Para esta práctica se utiliza Regresión Logística como modelo de clasificación, encapsulado dentro del Pipeline con StandardScaler. La Regresión Logística es un modelo lineal que, a pesar de su nombre, se utiliza para clasificación. En el caso multiclase, scikit-learn utiliza por defecto el esquema *one-vs-rest*, que entrena un clasificador binario por cada clase.

Tabla 5: Configuración del modelo

Componente	Configuración
Paso 1: StandardScaler	Estandarización Z-score
Paso 2: Regresión Logística	<code>max_iter=1000, random_state=42</code>

2.7. Métricas

Al tratarse de un problema de clasificación multiclase con 7 clases, la elección de métricas es crucial. Utilicé las siguientes:

- **Accuracy:** Proporción de predicciones correctas sobre el total. Es la métrica más intuitiva, pero puede ser engañosa si las clases están desbalanceadas (en nuestro caso no lo están significativamente, por lo que es una métrica válida).
- **Precision** (weighted): Mide cuántas de las predicciones positivas de cada clase son correctas, ponderado por el número de muestras de cada clase. Responde a: “de todos los que predije como Obesity_Type_I, ¿cuántos realmente lo son?”
- **Recall** (weighted): Mide cuántas muestras de cada clase fueron correctamente identificadas, ponderado por clase. Responde a: “de todos los que realmente son Obesity_Type_I, ¿cuántos identifiqué?”
- **F1-Score** (weighted): Media armónica de Precision y Recall, ponderada por clase. Es la métrica principal que utilizamos para optimización, ya que penaliza situaciones donde Precision y Recall son muy dispares.

Utilicé la variante **weighted** porque pondera cada clase por su frecuencia en el dataset, dando más peso a las clases con más muestras. Aun que en un dataset razonablemente balanceado como el nuestro, **weighted** y **macro** dan resultados similares.

3. Resultados

3.1. Modelo Base: Regresión Logística con Pipeline

El primer paso fue entrenar el modelo base (Pipeline con StandardScaler + Regresión Logística con parámetros por defecto) y evaluarlo directamente en el conjunto de test. Los resultados se muestran en la Tabla 6.

Tabla 6: Resultados del modelo base en el conjunto de test

Métrica	Valor
Accuracy	0.8910
Precision (weighted)	0.8901
Recall (weighted)	0.8910
F1-Score (weighted)	0.8897

Un accuracy de 0.891 para un problema de 7 clases es un resultado bastante bueno como punto de partida. Para contextualizar, un clasificador aleatorio obtendría aproximadamente $1/7 \approx 14.3\%$ de accuracy, por lo que estamos muy por encima de la línea base. El F1-Score (0.8897) es muy cercano al accuracy, lo que indica que el modelo no tiene un sesgo marcado hacia ninguna clase en particular.

El classification report detallado por clase revela información muy interesante:

Tabla 7: Classification Report del modelo base por clase

Clase	Precision	Recall	F1	Support
Insufficient_Weight	0.90	1.00	0.95	54
Normal_Weight	0.75	0.70	0.73	57
Obesity_Type_I	0.92	0.96	0.94	70
Obesity_Type_II	0.98	0.97	0.97	60
Obesity_Type_III	0.98	0.98	0.98	65
Overweight_Level_I	0.76	0.78	0.77	58
Overweight_Level_II	0.91	0.83	0.86	58
Weighted avg	0.89	0.89	0.89	422

Hay un patrón clarísimo: las clases extremas (Insufficient_Weight, Obesity_Type_II, Obesity_Type_III) tienen F1-Scores excelentes (0.95–0.98), mientras que las clases intermedias (Normal_Weight, Overweight_Level_I) tienen F1-Scores más bajos (0.73–0.77). Esto tiene todo el sentido:

las clases extremas están más separadas en el espacio de features (personas muy delgadas o con obesidad severa tienen características muy diferenciadas), mientras que las clases intermedias se solapan más entre sí. Esto nos lleva a la conclusión de que la frontera entre “peso normal” y “sobrepeso nivel I” es más difusa.

3.2. Validación Cruzada

Antes de evaluar en test, realicé una validación cruzada 5-Fold sobre el conjunto de entrenamiento para obtener una estimación robusta del rendimiento. La validación cruzada divide el train en 5 partes iguales (folds), entrena en 4 de ellos y evalúa en el restante, repitiendo el proceso 5 veces. Al estar dentro del Pipeline, la estandarización se recalcula en cada fold de forma independiente.

Los resultados fueron:

- **Accuracy CV:** $0,8635 \pm 0,0315$
- **F1-Score CV:** $0,8613 \pm 0,0317$

La desviación estándar de $\pm 0,031$ indica una estabilidad razonable del modelo entre folds. Que el resultado en test (0.891) sea ligeramente superior a la media de CV (0.864) no es preocupante: la validación cruzada se realiza sobre el train (que tiene menos datos por fold que el train completo), por lo que es normal que el rendimiento sea algo menor. Lo importante es que no hay una discrepancia enorme que sugiera overfitting.

3.3. Matriz de Confusión

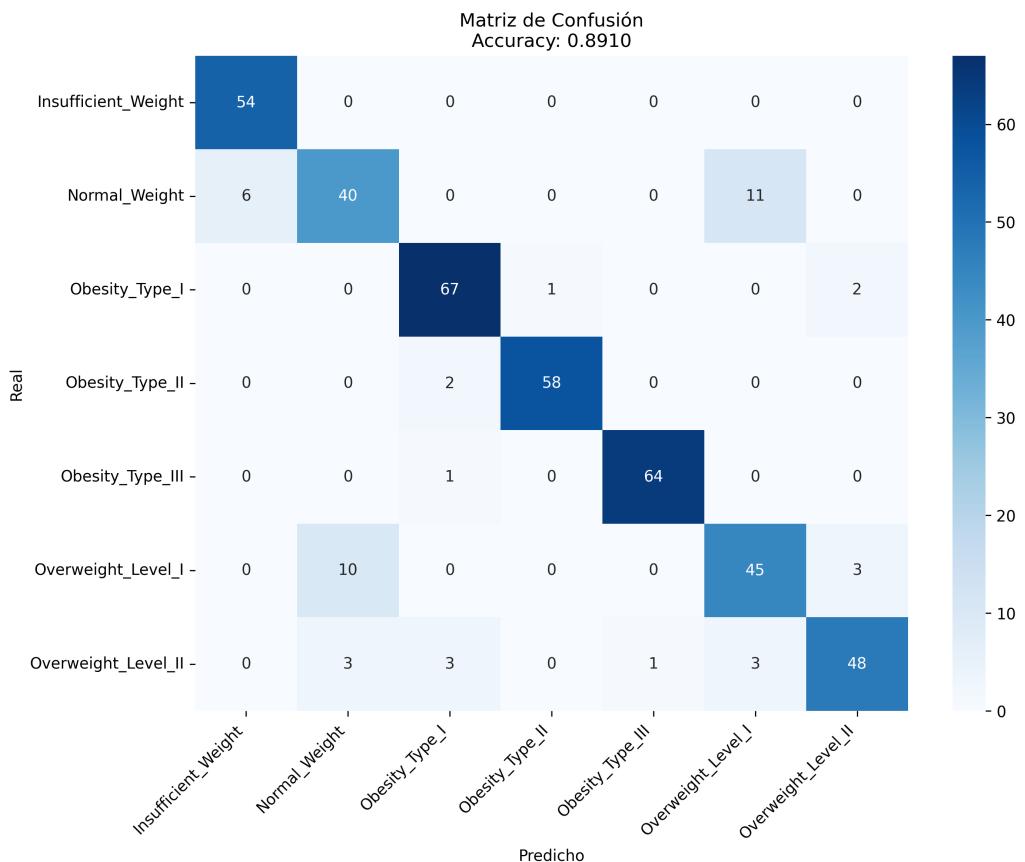


Figura 8: Matriz de confusión del modelo base (Pipeline con Regresión Logística, C=1.0).

La matriz de confusión (Figura 8) confirma lo observado en el classification report. La diagonal principal concentra la mayoría de predicciones. Los errores se concentran en las clases intermedias: Normal_Weight se confunde principalmente con Overweight_Level_I (y viceversa), y Overweight_Level_II se confunde parcialmente con Obesity_Type_I. Esto es coherente con la realidad

clínica, donde las fronteras entre categorías de IMC son umbrales arbitrarios y personas cercanas al umbral pueden ser clasificadas en una u otra categoría.

En cambio, las clases extremas (Insufficient_Weight con recall = 1.00, Obesity_Type_III con recall = 0.98) son identificadas prácticamente sin error. Ninguna persona con peso insuficiente fue clasificada erróneamente, y solo 1 de 65 personas con obesidad tipo III fue mal clasificada.

3.4. Optimización de Hiperparámetros con GridSearchCV

Para intentar mejorar los resultados del modelo base, se aplicó GridSearchCV para buscar la mejor combinación de hiperparámetros. GridSearchCV prueba exhaustivamente todas las combinaciones de un grid de parámetros, evaluando cada una mediante validación cruzada 5-Fold interna.

El hiperparámetro más importante de la Regresión Logística es **C** (inversa de la fuerza de regularización):

- **C pequeño** (ej: 0.01): Regularización fuerte, modelo más simple, menos riesgo de overfitting pero posible underfitting.
- **C grande** (ej: 10): Regularización débil, modelo más complejo, mejor ajuste a los datos de entrenamiento pero más riesgo de overfitting.

El grid de búsqueda fue:

- `model__C`: [0.01, 0.1, 1, 10]
- `model__max_iter`: [1000, 2000]

Los resultados de la optimización se resumen en la Tabla 8.

Tabla 8: Resultados de GridSearchCV

Parámetro	Valor
Mejor C	10
Mejor max_iter	1000
Mejor F1-Score CV	0.9279

El mejor valor encontrado fue **C = 10**, lo que indica que el modelo se beneficia de una regularización más débil. Esto tiene sentido: con 23 features y 1685 muestras de entrenamiento, tenemos suficientes datos para que un modelo más complejo no sobreajuste.

Al evaluar el modelo optimizado en el conjunto de test, obtuvimos una mejora significativa:

Tabla 9: Comparación modelo base vs modelo optimizado en test

Métrica	Base (C=1)	Optimizado (C=10)
Accuracy	0.8910	0.9360
Precision	0.8901	0.9365
Recall	0.8910	0.9360
F1-Score	0.8897	0.9359

Se obtubo una mejora de +4.5 puntos porcentuales en accuracy y +4.6 puntos en F1-Score. Para un problema de 7 clases, pasar de 0.89 a 0.94 es un salto muy significativo que demuestra la importancia de la optimización de hiperparámetros.

El classification report del modelo optimizado muestra mejoras generalizadas en todas las clases:

Tabla 10: Classification Report del modelo optimizado ($C=10$) por clase

Clase	Precision	Recall	F1	Support
Insufficient_Weight	0.95	1.00	0.97	54
Normal_Weight	0.86	0.84	0.85	57
Obesity_Type_I	0.94	0.97	0.96	70
Obesity_Type_II	0.97	0.97	0.97	60
Obesity_Type_III	1.00	0.98	0.99	65
Overweight_Level_I	0.85	0.86	0.85	58
Overweight_Level_II	0.98	0.91	0.95	58
Weighted avg	0.94	0.94	0.94	422

Las mejoras más destacadas se producen exactamente donde el modelo base fallaba más:

- **Normal_Weight**: F1 sube de 0.73 a 0.85 (+12 puntos)
- **Overweight_Level_I**: F1 sube de 0.77 a 0.85 (+8 puntos)
- **Overweight_Level_II**: F1 sube de 0.86 a 0.95 (+9 puntos)
- **Obesity_Type_III**: F1 sube de 0.98 a 0.99, con precision perfecta de 1.00

Es decir, al reducir la regularización ($C = 10$ vs $C = 1$), el modelo consigue trazar fronteras de decisión más precisas entre las clases intermedias, que es exactamente donde más lo necesitaba. Esto confirma que el modelo base estaba sufriendo de underfitting: la regularización por defecto ($C = 1$) era demasiado restrictiva para la complejidad del problema.

4. Análisis y Discusión

4.1. Impacto del Preprocesamiento

El preprocesamiento fue, sin duda, la fase que más tiempo consumió pero también la que más impacto tuvo en la calidad final del modelo. Quiero destacar varios puntos:

La transformación Box-Cox de Age fue clave. Sin ella, habría perdido 168 muestras (8 % del dataset) por outliers. Gracias a la transformación, solo eliminé 2 muestras adicionales, preservando la información de personas mayores que, aunque menos frecuentes, representan casos legítimos y valiosos para el aprendizaje.

La decisión de no tocar NCP también fue clave. Ante los 579 “outliers” detectados por IQR (27.43 %), la tentación era aplicar alguna transformación o eliminar datos. Sin embargo, el entender que esos valores atípicos eran artefactos de SMOTE (y no datos erróneos) me permitió tomar y argumentar la decisión de conservarlos. Esto ilustra la importancia de entender los datos antes de transformarlos: un análisis ciego habría eliminado un cuarto del dataset sin justificación real.

El Pipeline eliminó el riesgo de Data Leakage. Al encapsular la estandarización dentro del Pipeline, se garantizó que en cada fold de la validación cruzada y en cada combinación de GridSearchCV, el scaler se ajustase únicamente a los datos de entrenamiento de ese fold. Sin Pipeline, habría sido fácil cometer el error de escalar todo el dataset antes de dividir.

4.2. Análisis del Modelo

La Regresión Logística, a pesar de ser un modelo lineal, alcanza un F1 de 0.9359 en un problema de 7 clases. Esto es especialmente notable considerando que el PCA mostró que los datos no son linealmente separables en 2D. La explicación es doble: primero, la Regresión Logística opera en el espacio de 23 dimensiones (no en 2D como el PCA), donde las fronteras lineales son mucho más expresivas y segundo, la optimización de $C = 10$ permite al modelo trazar fronteras más complejas al relajar la regularización.

El patrón de errores es muy coherente: las confusiones se concentran entre clases adyacentes en la escala de IMC. Normal_Weight se confunde con Overweight_Level_I, Overweight_Level_I con Overweight_Level_II, etc. Esto refleja la naturaleza continua del peso corporal: la frontera entre “peso normal” y “sobrepeso leve” es un umbral artificial ($IMC = 25$), y personas cercanas a ese umbral pueden caer en cualquiera de las dos categorías. En un contexto clínico, estas confusiones entre clases adyacentes son mucho menos graves que confundir, por ejemplo, Insufficient_Weight con Obesity_Type_III.

4.3. Efecto de la Optimización

La mejora de +4.6 puntos en F1-Score mediante GridSearchCV demuestra que la selección de hiperparámetros no es un detalle menor, sino una parte esencial del flujo de ML. El paso de $C = 1$ a $C = 10$ implica que el modelo por defecto estaba subregularizado para los datos que tiene: con 1685 muestras y 23 features, hay suficiente información para permitir un modelo más expresivo sin caer en overfitting.

El hecho de que $\text{max_iter} = 1000$ fuese suficiente (no se necesitó 2000) confirma que el modelo converge correctamente y no hay problemas numéricos de optimización.

5. Conclusiones

En este trabajo he implementado un flujo completo de Machine Learning, desde la exploración de datos hasta la optimización del modelo, aplicado a la predicción de niveles de obesidad. Las principales conclusiones son:

1. **El preprocessamiento inteligente preserva información valiosa.** La transformación Box-Cox sobre Age permitió reducir la pérdida de datos del 8 % al 0.1 % (de 168 a 2 filas eliminadas), demostrando que las transformaciones estadísticas son preferibles a la eliminación directa cuando hay muchos outliers legítimos.
2. **No todos los outliers son realmente outliers.** El caso de NCP (579 valores atípicos por IQR) demuestra la importancia de entender el origen de los datos antes de tomar decisiones de preprocessamiento. Los artefactos de SMOTE no son errores: son consecuencia del proceso de generación sintética.
3. **Los Pipelines son imprescindibles.** Encapsular la estandarización y el modelo en un Pipeline elimina por completo el riesgo de Data Leakage y simplifica el código enormemente, especialmente al combinarse con validación cruzada y GridSearchCV.
4. **La optimización de hiperparámetros marca la diferencia.** Pasar de $C = 1$ (por defecto) a $C = 10$ (optimizado) mejoró el F1-Score de 0.8897 a 0.9359, una ganancia de +4.6 puntos que se tradujo en mejoras generalizadas en todas las clases, especialmente las intermedias.
5. **Las clases extremas son más fáciles que las intermedias.** El modelo alcanza $F1 > 0.95$ en las clases extremas (Insufficient_Weight, Obesity_Type_III) pero solo 0.85 en las intermedias (Normal_Weight, Overweight_Level_I). Esto refleja la naturaleza continua del peso corporal y la arbitrariedad de los umbrales de clasificación.