Práctica 1: Aprendizaje Supervisado

Comparativa de SVM, Árboles y Ensembles sobre el dataset de Obesidad

Jordi Blasco Lozano
DNI: 74527208D
Universidad de Alicante - Escuela Politécnica Superior
Aprendizaje Avanzado - Curso 2025/2026
Email: jbl42@alu.ua.es

Resumen

En esta práctica se comparan distintos algoritmos de aprendizaje supervisado sobre el mismo problema de clasificación multiclase: la estimación del nivel de obesidad a partir de hábitos alimenticios y condición física. Se reutiliza el dataset y el preprocesado de la Práctica 0 para garantizar comparabilidad, y se evalúan modelos SVM (distintos kernels y ajuste de hiperparámetros), árboles de decisión con técnicas de poda, y métodos de ensemble (Random Forest, Extra Trees, Gradient Boosting y AdaBoost). Los resultados se reportan mediante validación cruzada 5-Fold, métricas ponderadas (Precision/Recall/F1) y matrices de confusión del mejor modelo de cada bloque, concluyendo con una comparativa global.

1. Introducción

El objetivo principal de esta práctica es aplicar y comparar algoritmos clásicos de aprendizaje supervisado vistos en teoría sobre un problema real de clasificación multiclase. Para facilitar la comparación con la práctica anterior, se utiliza el mismo dataset de la Práctica 0: *Estimation of Obesity Levels Based on Eating Habits and Physical Condition* (UCI ML Repository), con 2111 instancias y 17 variables (16 características + 1 variable objetivo con 7 clases).

2. Configuración Experimental

2.1. Dataset

El dataset contiene variables numéricas (edad, altura, peso, frecuencia de actividad, etc.) y categóricas (hábitos de comida, consumo de alcohol, medio de transporte, etc.). La Tabla 1 resume sus características.

Tabla 1: Características del dataset de Obesidad

Característica	Valor
Filas	2111
Columnas	17 (16 features + 1 target)
Variable objetivo	NObeyesdad (7 clases)
Tipo de problema	Clasificación multiclase
Valores faltantes	0

2.2. Preprocesado y partición

Siguiendo el enunciado, el preprocesado se da por supuesto y no se vuelve a detallar exhaustivamente. En cualquier caso, para mantener consistencia con la Práctica 0 se aplica exactamente la

misma receta: (i) filtrado IQR de outliers en Weight/Height, (ii) transformación Box-Cox de Age con filtrado IQR posterior, (iii) encoding de variables categóricas (LabelEncoding para binarias y One-Hot para multiclase con `drop_first`), (iv) split train/test estratificado 80/20, y (v) filtrado adicional de outliers en train por Z-score ($|z|>3$) en variables numéricas originales.

2.3. Métricas

Para cada configuración se reporta Accuracy en validación cruzada 5-Fold (CV Accuracy) y, sobre test, Accuracy y métricas ponderadas (Precision/Recall/F1 weighted) para evitar sesgos cuando hay diferencias de frecuencia entre clases. Además, se incluyen matrices de confusión del mejor modelo de cada bloque y una matriz final del mejor modelo global.

3. Resultados

3.1. Parte 1: Support Vector Machines

La Tabla 2 recoge la comparación inicial de SVM con diferentes kernels usando hiperparámetros por defecto. A partir de estos resultados, se profundiza en la optimización de RBF (C y γ) y del kernel polinomial (degree, C y γ).

Tabla 2: Comparativa SVM con kernels por defecto (CV y test).

Kernel	CV Accuracy	Test Accuracy	Precision (w)	Recall (w)	F1 (w)	Nº SV	% Train SV
Linear	0.9418	0.9384	0.9391	0.9384	0.9381	565	33.5312
RBF	0.8528	0.8720	0.8807	0.8720	0.8747	1089	64.6291
Poly (d=3)	0.7454	0.7441	0.7612	0.7441	0.7389	1202	71.3353
Sigmoid	0.6409	0.6517	0.6495	0.6517	0.6501	1217	72.2255

Tabla 3: Optimización de SVM-RBF: combinaciones representativas de C y γ.

C	gamma	CV Accuracy	Test Accuracy	Nº SV	% Train	Tiempo (s)
100.0000	0.01	0.9401	0.9431	629	37.3294	0.0426
100.0000	0.001	0.9223	0.9408	879	52.1662	0.0475
100.0000	scale	0.9086	0.9218	799	47.4184	0.0473
100.0000	auto	0.9086	0.9218	799	47.4184	0.0457
10.0000	scale	0.9080	0.9100	850	50.4451	0.0462
10.0000	auto	0.9080	0.9100	850	50.4451	0.0474
10.0000	0.01	0.9074	0.9194	880	52.2255	0.0409
10.0000	0.1	0.8896	0.8981	943	55.9644	0.0643
100.0000	0.1	0.8890	0.9028	939	55.7270	0.0638
1.0000	0.1	0.8659	0.8602	1053	62.4926	0.0601

Tabla 4: Optimización de SVM-Poly: combinaciones representativas de degree, C y γ.

degree	C	gamma	CV Accuracy	Test Accuracy	Nº SV
3	10.0000	0.1	0.8718	0.8886	789
3	1.0000	0.1	0.8641	0.8768	897
3	10.0000	auto	0.8605	0.8791	916
3	10.0000	scale	0.8605	0.8791	916
2	10.0000	0.1	0.8588	0.8460	783
2	10.0000	scale	0.8475	0.8555	900
2	10.0000	auto	0.8475	0.8555	900
4	1.0000	0.1	0.8392	0.8436	959
4	10.0000	0.1	0.8356	0.8460	856
2	1.0000	0.1	0.8309	0.8389	946

También se comparan implementaciones del caso lineal (SVC lineal, LinearSVC y SGDClassifier con hinge loss), destacando diferencias de coste computacional y formulación.

Tabla 5: Comparación de implementaciones lineales de SVM.

Implementación	CV Accuracy	Test Accuracy	Nº SV	Tiempo (s)
SVC(linear)	0.9418	0.9384	565	0.0581
LinearSVC	0.7632	0.7867	-	0.0471
SGDClassifier	0.7062	0.6943	-	0.0274

El análisis de vectores de soporte ayuda a interpretar la complejidad efectiva del clasificador: a mayor número de SV, típicamente mayor frontera efectiva (y mayor coste en inferencia para SVC).

Tabla 6: Análisis de vectores de soporte en configuraciones representativas.

Configuración	Nº SV	% Training Set
Linear (C=1)	565	33.5312
RBF (C=10, =0.1)	943	55.9644
RBF (C=1, =scale)	1089	64.6291
Poly (d=3, C=1)	1202	71.3353
RBF (C=100, =1)	1260	74.7774

Finalmente, se evalúa el mejor modelo SVM (según GridSearch sobre RBF) y se incluye su matriz de confusión en la Figura 1.

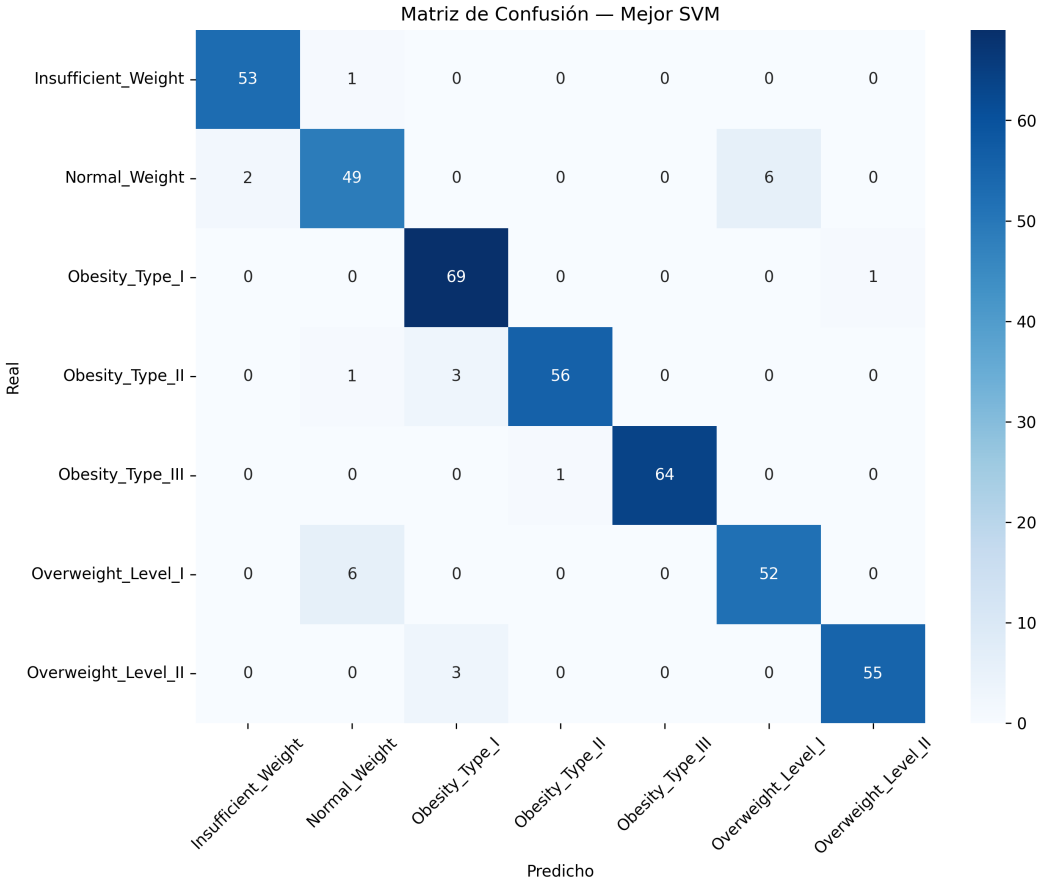

Figura 1: Matriz de confusión del mejor modelo SVM.

Tabla 7: Resumen de la Parte 1 (SVM): mejor configuración por kernel.

Modelo	CV Accuracy	Test Accuracy	F1 (w)	Nº SV	% Train SV
SVM Linear	0.9418	0.9384	0.9381	565	33.5312
SVM RBF (best)	0.9401	0.9431	0.9432	629	37.3294
SVM Poly (best)	0.8718	0.8886	0.8885	789	46.8249
SVM Sigmoid	0.6409	0.6517	0.6501	1217	72.2255

El kernel lineal resulta el más eficaz con parámetros por defecto (CV = 0.9418), gracias a que el dataset, tras One-Hot Encoding y estandarización, es relativamente separable en el espacio de 23 dimensiones. El kernel RBF, tras optimización (C = 100, $\gamma = 0.01$), alcanza prácticamente la misma accuracy (Test = 0.9431). El kernel polinomial queda algo por debajo incluso después de GridSearch (Test = 0.8886), lo que sugiere que la frontera de decisión real no tiene una estructura polinomial clara. El kernel sigmoide no resulta competitivo en absoluto.

La comparación de implementaciones lineales confirma que `SVC(kernel='linear')` y `LinearSVC` dan prácticamente el mismo resultado, pero `SGDClassifier` queda por debajo al converger a un óptimo diferente mediante descenso de gradiente estocástico.

3.2. Parte 2: Árboles de Decisión

Se parte de un árbol sin restricciones para observar sobreajuste (train accuracy muy alto frente a test). Después se explora el efecto de `max_depth`, los criterios de impureza y técnicas de poda previa y posterior.

Tabla 8: Baseline de Decision Tree sin restricciones.

Configuración	Train Accuracy	Test Accuracy	Profundidad	Nº Hojas	Tiempo (s)
Sin restricciones	1.0000	0.9289	12	101	0.0085

Tabla 9: Efecto de `max_depth` en Decision Tree.

max_depth	Train Acc	CV Acc	Test Acc	Profundidad real	Nº Hojas
1	0.2908	0.2908	0.2915	1	2
2	0.5543	0.5531	0.5450	2	4
3	0.6433	0.6291	0.6564	3	8
5	0.8528	0.8231	0.8199	5	25
7	0.9454	0.8855	0.9005	7	57
10	0.9988	0.9252	0.9289	10	98
15	1.0000	0.9228	0.9289	12	101
None	1.0000	0.9240	0.9289	12	101

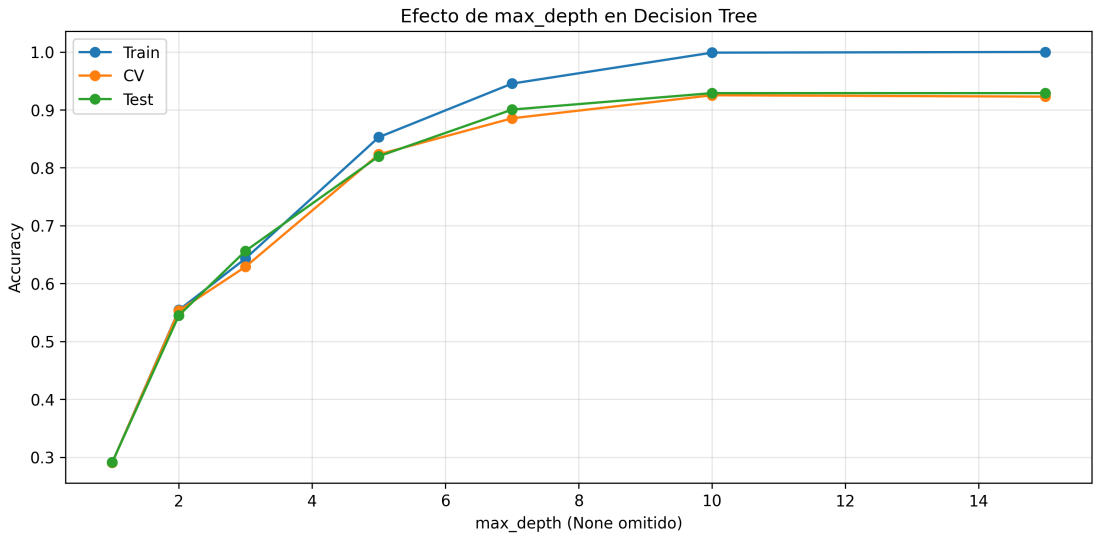

Figura 2: Curva de rendimiento (train/CV/test) en función de `max_depth`.

Tabla 10: Comparación de criterios de división (gini, entropy, log_loss).

Criterio	CV Accuracy	Test Accuracy	Nº Hojas
entropy	0.9418	0.9597	79
log_loss	0.9418	0.9597	79
gini	0.9240	0.9289	101

Tabla 11: Poda previa: configuraciones representativas encontradas por GridSearch.

max_depth	min_samples_split	min_samples_leaf	max_leaf_nodes	CV Acc	Test Acc
None	2	2	None	0.9288	0.9218
None	5	1	None	0.9276	0.9289
None	5	2	None	0.9270	0.9171
10	2	2	None	0.9252	0.9218
10	2	1	None	0.9252	0.9289
10	5	1	None	0.9252	0.9289
10	5	2	None	0.9240	0.9218
None	2	1	None	0.9240	0.9289
None	2	2	50	0.9228	0.9265
10	2	2	50	0.9223	0.9265

Tabla 12: Poda posterior por cost-complexity pruning (ccp_alpha).

ccp_alpha	CV Accuracy	Test Accuracy	Profundidad	Nº Hojas
0.0008	0.9276	0.9289	10	74
0.0011	0.9264	0.9265	10	57
0.0014	0.9252	0.9242	10	49
0.0011	0.9252	0.9265	10	67
0.0000	0.9240	0.9289	12	101
0.0006	0.9240	0.9289	10	86
0.0021	0.9181	0.9360	9	41
0.0024	0.9151	0.9289	8	35
0.0043	0.8991	0.9242	8	28
0.0084	0.8724	0.9005	8	22
0.0214	0.7875	0.8009	6	12
0.0912	0.5027	0.4313	2	3

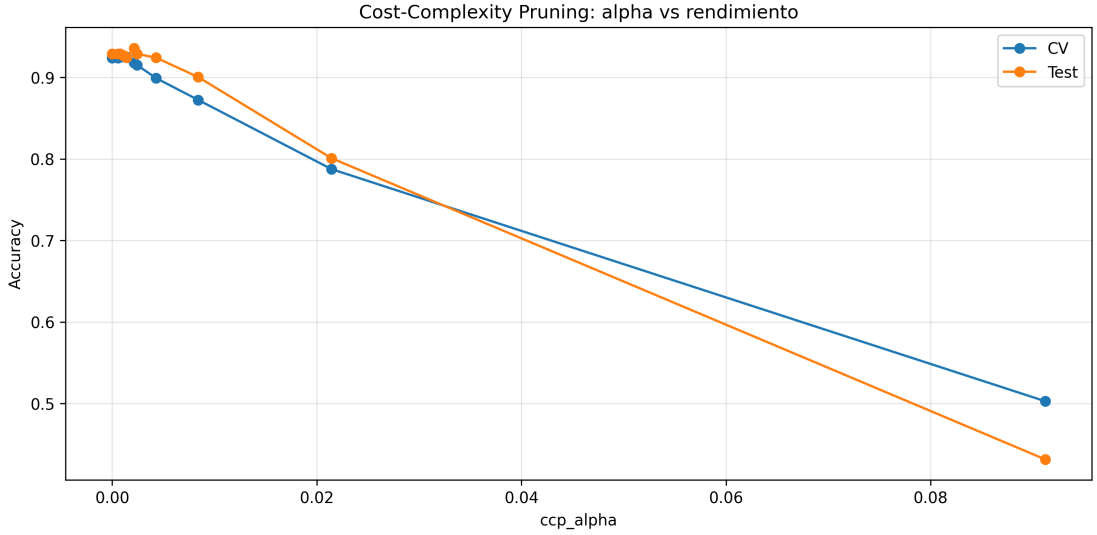

Figura 3: Cost-complexity pruning: efecto de `ccp_alpha` en CV/test.

Tabla 13: Importancia de características del mejor árbol (top).

Feature	Importancia (Gini)
Weight	0.5066
Height	0.2470
Gender	0.1603
Age	0.0248
FAVC	0.0238
CH2O	0.0101
CAEC_Sometimes	0.0076
TUE	0.0047
NCP	0.0038
CAEC_Frequently	0.0023
MTRANS_Walking	0.0023
SCC	0.0018
CALC_no	0.0016
MTRANS_Public_Transportation	0.0011
FAF	0.0008
family_history_with_overweight	0.0007
SMOKE	0.0005
CALC_Sometimes	0.0004
FCVC	0.0000
CAEC_no	0.0000

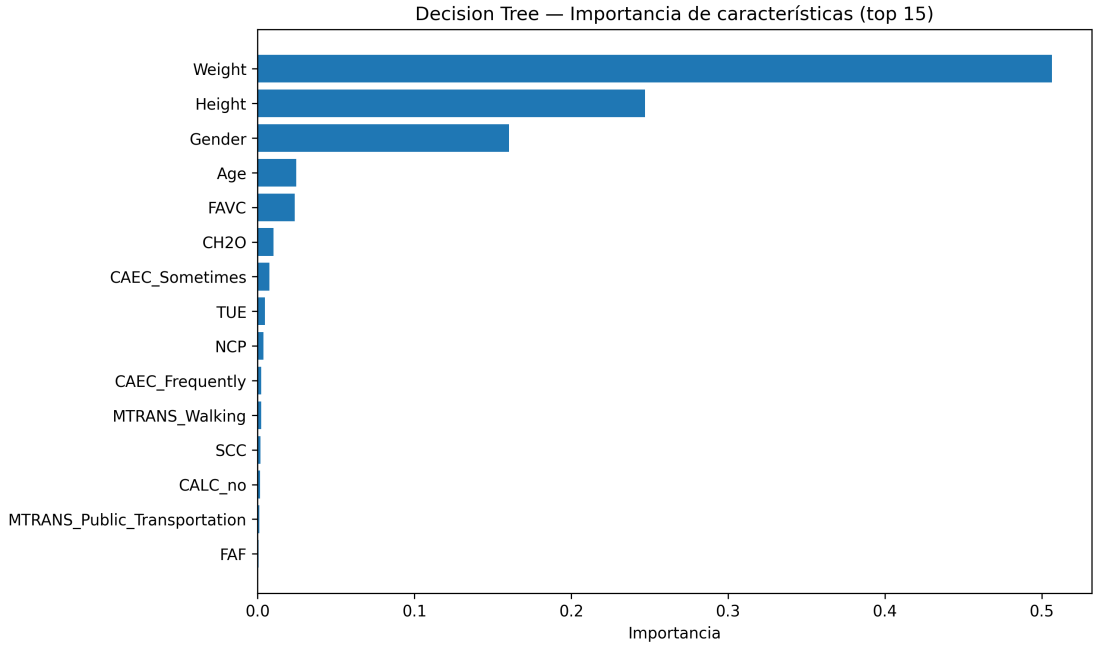

Figura 4: Importancia de características del árbol (top 15).

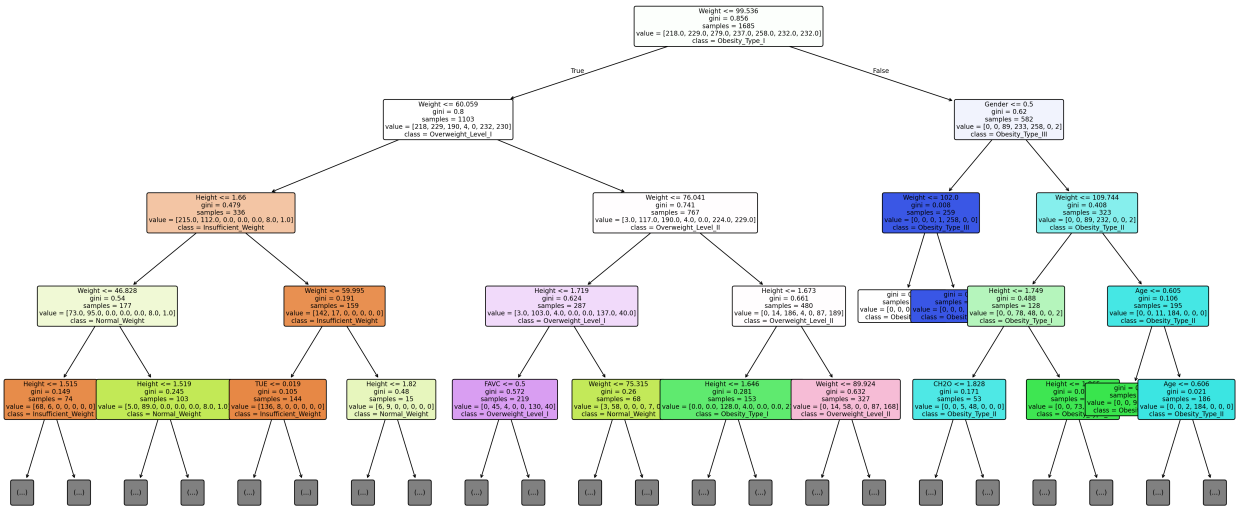

Figura 5: Visualización del árbol (profundidad limitada para legibilidad).

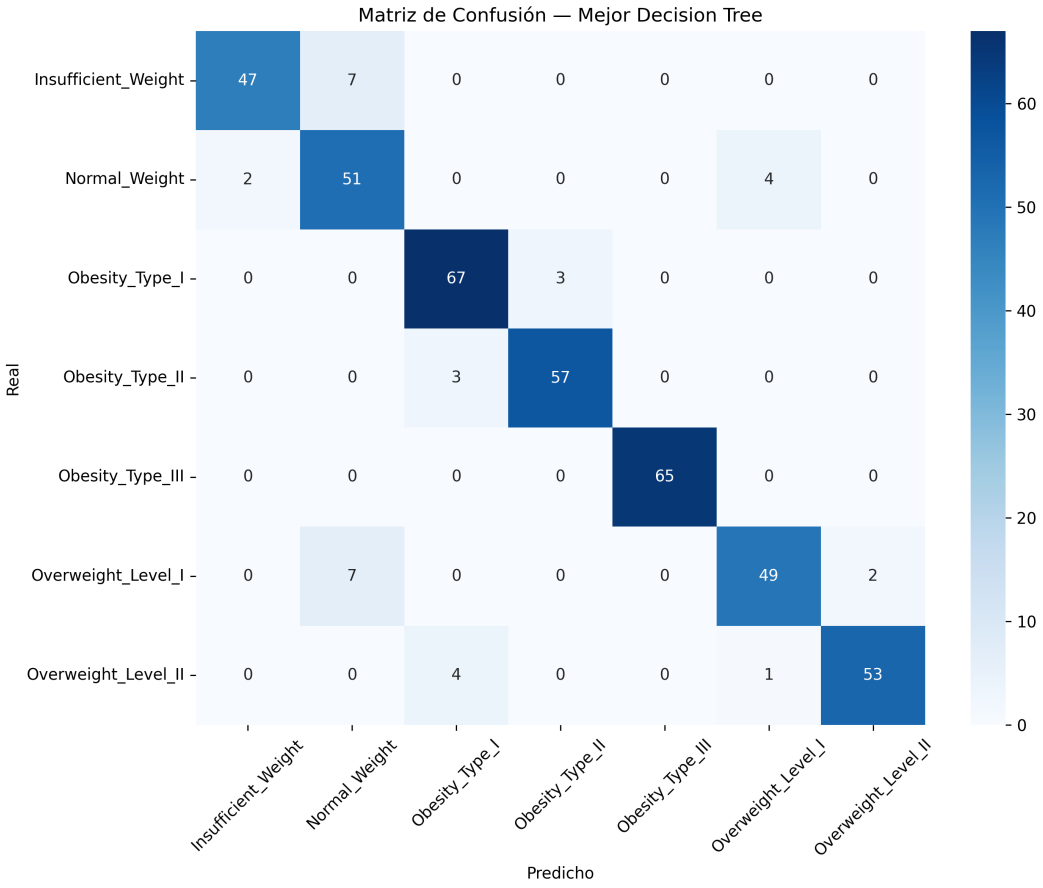

Figura 6: Matriz de confusión del mejor Decision Tree.

Tabla 14: Resumen de la Parte 2 (Árboles): baseline vs poda previa vs poda posterior.

Modelo	Test Accuracy	Precision (w)	Recall (w)	F1 (w)	Profundidad	Nº Hojas
Sin restricciones	0.9289	0.9327	0.9289	0.9297	12	101
Mejor poda posterior	0.9289	0.9327	0.9289	0.9297	10	74
Mejor poda previa	0.9218	0.9252	0.9218	0.9224	11	90

El árbol sin restricciones alcanza 100 % en train pero solo 92.89 % en test (profundidad 12, 101 hojas), síntoma claro de sobreajuste moderado. La poda posterior mediante cost-complexity pruning (ccp_alpha ≈ 0.0008) logra el mismo rendimiento en test con un árbol más compacto (profundidad 10, 74 hojas), es decir, elimina ramas redundantes sin perder capacidad predictiva. La poda previa por GridSearch queda ligeramente por debajo (92.18 %), probablemente porque las restricciones de `min_samples_leaf` impiden capturar algunas divisiones finas legítimas.

La importancia de características confirma que **Weight** y **Height** dominan la predicción del nivel de obesidad, coherente con la relación directa entre estas variables y el IMC.

3.3. Parte 3: Random Forest y Extra Trees

Los métodos de ensemble basados en bagging reducen varianza y mejoran la estabilidad respecto a un único árbol. Se evalúa un Random Forest por defecto, el efecto de `n_estimators` y `max_features`, un GridSearch de hiperparámetros y la comparación con Extra Trees.

Tabla 15: Random Forest por defecto.

Configuración	OOB Score	Test Accuracy	F1 (w)	Tiempo (s)
RF por defecto (n=100)	0.9412	0.9455	0.9468	0.1387

Tabla 16: Efecto de `n_estimators` en OOB y test.

n_estimators	OOB Score	Test Accuracy	Tiempo (s)
10	0.8469	0.9100	0.0365
25	0.9086	0.9431	0.0520
50	0.9276	0.9479	0.0745
100	0.9412	0.9455	0.1494
200	0.9442	0.9479	0.2710
500	0.9472	0.9526	0.6264

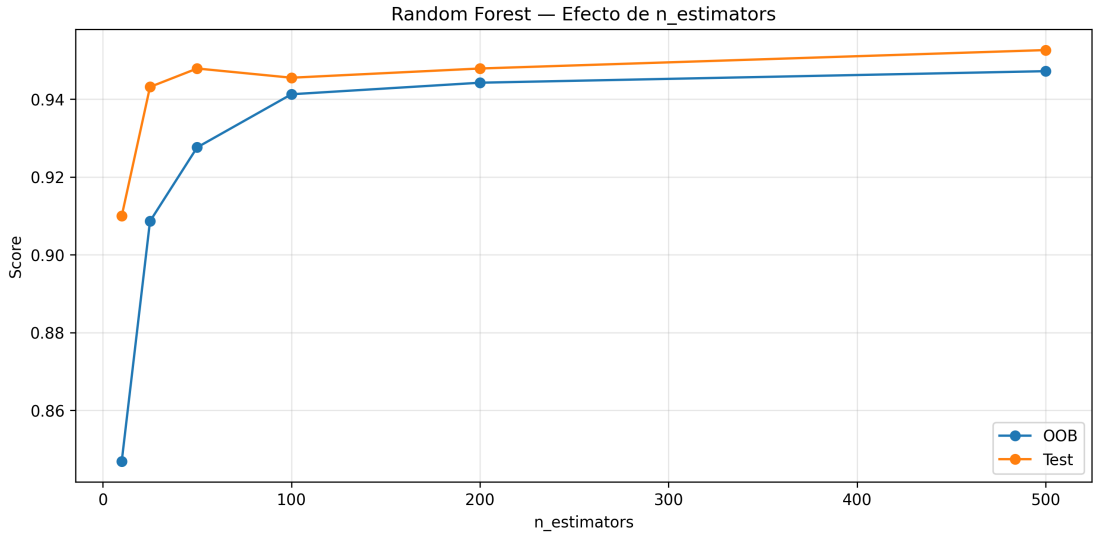

Figura 7: Efecto de `n_estimators`: OOB y test.

Tabla 17: Efecto de `max_features` (n=100).

max_features	OOB Score	Test Accuracy
0.5	0.9596	0.9550
None	0.9543	0.9526
log2	0.9412	0.9455
sqrt	0.9412	0.9455

Tabla 18: GridSearch: configuraciones representativas.

n_estimators	max_features	max_depth	min_samples_leaf	CV Acc	Test Acc
200	None	None	1	0.9519	0.9550
200	None	20	1	0.9519	0.9550
100	None	20	1	0.9507	0.9526
100	None	None	1	0.9507	0.9526
200	None	10	1	0.9496	0.9526
200	None	None	2	0.9484	0.9573
200	None	20	2	0.9484	0.9573
100	None	10	1	0.9478	0.9573
100	None	20	2	0.9472	0.9597
100	None	None	2	0.9472	0.9597

Tabla 19: Importancia de características en Random Forest (Gini vs Permutation).

Feature	Gini Importance	Permutation Importance	Perm Std
Weight	0.4585	0.7384	0.0187
Height	0.2463	0.3116	0.0154
Gender	0.1583	0.1720	0.0139
Age	0.0388	0.0581	0.0076
FAVC	0.0235	0.0057	0.0022
FAF	0.0128	-0.0021	0.0027
FCVC	0.0099	-0.0005	0.0026
CH2O	0.0089	0.0028	0.0028
TUE	0.0086	0.0021	0.0029
NCP	0.0086	0.0045	0.0027
CALC_no	0.0047	-0.0019	0.0014
CAEC_Frequently	0.0045	-0.0026	0.0022
CAEC_Sometimes	0.0038	-0.0045	0.0017
CALC_Sometimes	0.0032	0.0005	0.0009
family_history_with_overweight	0.0024	0.0012	0.0016
MTRANS_Public_Transportation	0.0017	0.0000	0.0015
MTRANS_Walking	0.0015	-0.0024	0.0000
SMOKE	0.0013	0.0000	0.0000
SCC	0.0011	-0.0002	0.0007
CALC_Frequently	0.0006	0.0000	0.0000
MTRANS_Motorbike	0.0005	0.0000	0.0000
CAEC_no	0.0005	0.0000	0.0000
MTRANS_Bike	0.0001	0.0000	0.0000

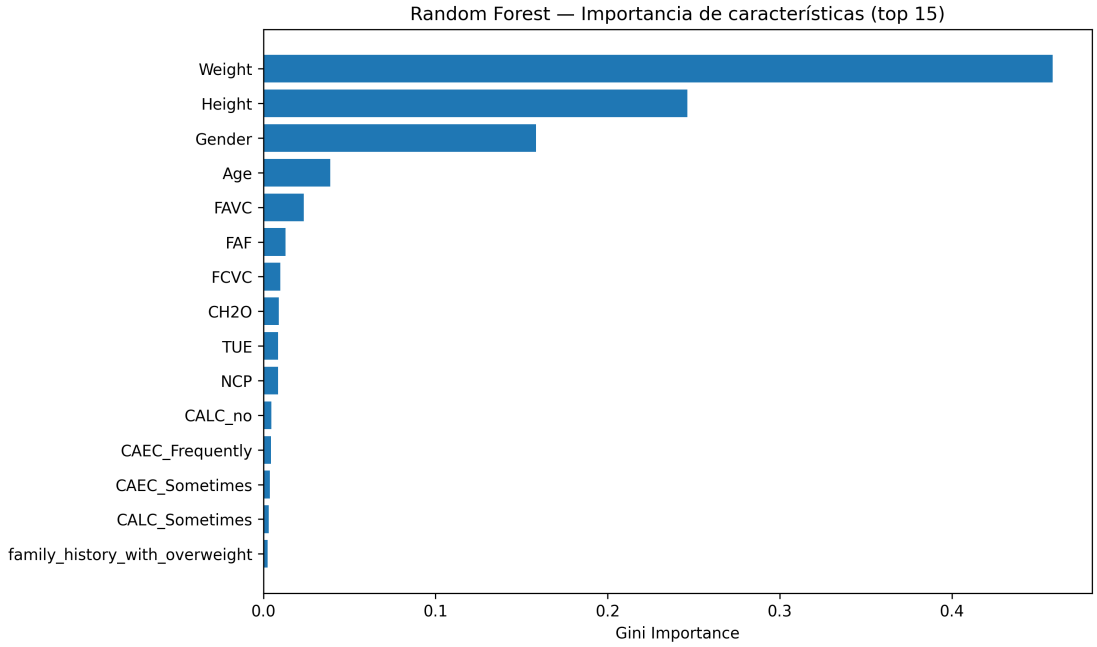

Figura 8: Importancia de características (top 15, Gini) en Random Forest.

Tabla 20: Resultados de Extra Trees.

Modelo	CV Accuracy	Test Accuracy	F1 (w)	Tiempo (s)
Extra Trees (n=100)	0.9163	0.9265	0.9277	0.0871

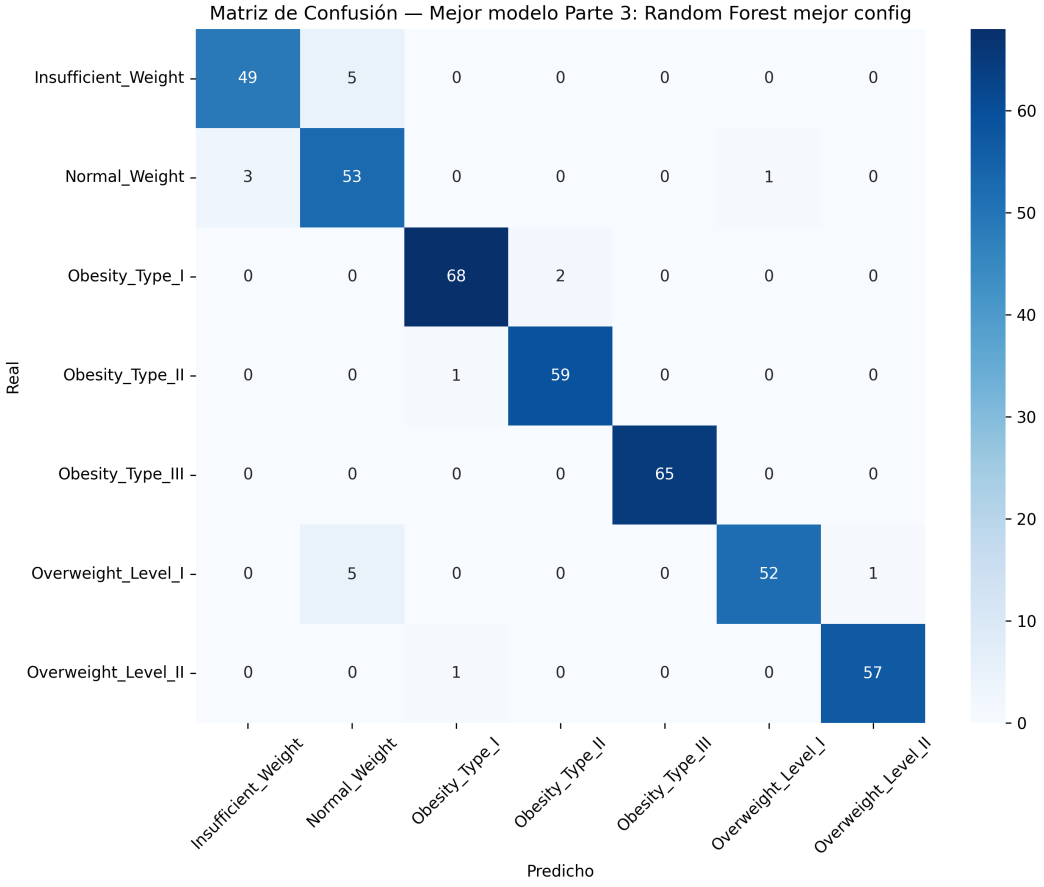

Figura 9: Matriz de confusión del mejor modelo de la Parte 3.

Tabla 21: Resumen de la Parte 3 (Ensembles bagging).

Modelo	Test Accuracy	Precision (w)	Recall (w)	F1 (w)	OOB Score	Tiempo (s)
Random Forest mejor config	0.9550	0.9568	0.9550	0.9553	-	-
Random Forest por defecto	0.9455	0.9527	0.9455	0.9468	0.9412	-
Decision Tree (baseline)	0.9289	0.9327	0.9289	0.9297	-	-
Extra Trees	0.9265	0.9325	0.9265	0.9277	-	-

El Random Forest por defecto (OOB = 0.9412, Test = 0.9455) ya supera ampliamente al mejor árbol individual, demostrando el beneficio de la agregación. Tras GridSearch, la mejor configuración (200 árboles, todas las features, sin límite de profundidad) alcanza Test = 0.9550. Un resultado interesante es que usar **todas las features** (`max_features=None`) supera a las opciones clásicas (`sqrt`, `log2`), sugiriendo que con solo 23 features y dos muy dominantes (Weight, Height), limitar las features por split reduce la calidad de los árboles más de lo que gana por decorrelación.

Extra Trees (Test = 0.9265) rinde por debajo del Random Forest estándar. La aleatorización extra en los umbrales no aporta beneficio aquí, probablemente porque el dataset no es tan ruidoso como para que merezca la pena el aumento de sesgo.

3.4. Parte 4: Gradient Boosting y AdaBoost

Los métodos de boosting suelen capturar relaciones no lineales mediante la suma secuencial de modelos débiles. Se evalúa Gradient Boosting por defecto, se estudia la relación entre `n_estimators` y `learning_rate`, se explora `max_depth` y se realiza GridSearch. También se evalúa AdaBoost (stumps).

Tabla 22: Gradient Boosting por defecto.

Modelo	CV Accuracy	Test Accuracy	F1 (w)	Tiempo (s)
Gradient Boosting (default)	0.9573	0.9668	0.9674	2.0868

Tabla 23: Efecto de `n_estimators` y `learning_rate`.

n_estimators	learning_rate	CV Accuracy	Test Accuracy
200	0.1000	0.9596	0.9716
50	0.5000	0.9579	0.9550
100	0.1000	0.9573	0.9668
50	0.1000	0.9454	0.9431
100	0.0500	0.9430	0.9455
200	0.0100	0.9068	0.9005

Tabla 24: Efecto de `max_depth` en Gradient Boosting.

max_depth	Train Acc	CV Accuracy	Test Accuracy
3	1.0000	0.9573	0.9668
5	1.0000	0.9561	0.9597
2	0.9905	0.9389	0.9408
1	0.8789	0.8439	0.8412

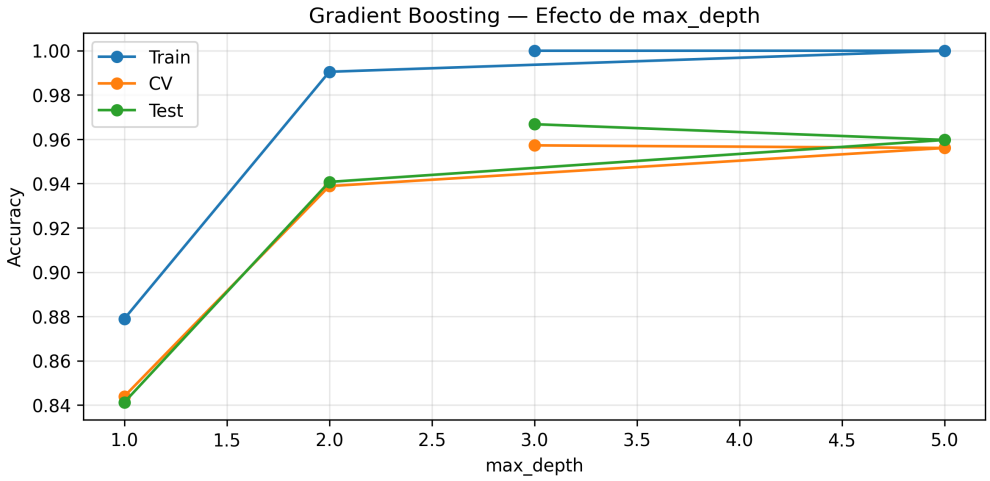

Figura 10: Efecto de `max_depth` en Gradient Boosting (train/CV/test).

Tabla 25: GridSearch: configuraciones representativas para Gradient Boosting.

n_estimators	learning_rate	max_depth	subsample	CV Acc	Test Acc
200	0.1000	3	1.0000	0.9596	0.9716
200	0.5000	3	0.8000	0.9585	0.9573
200	0.5000	3	1.0000	0.9585	0.9621
200	0.1000	3	0.8000	0.9585	0.9621
100	0.5000	3	0.8000	0.9579	0.9692
200	0.0500	3	1.0000	0.9579	0.9597
200	0.0500	3	0.8000	0.9573	0.9455
100	0.1000	3	1.0000	0.9573	0.9668
100	0.5000	3	1.0000	0.9567	0.9621
100	0.1000	3	0.8000	0.9561	0.9526

Tabla 26: Exploración de hiperparámetros en AdaBoost (stumps).

n_estimators	learning_rate	CV Accuracy	Test Accuracy	Tiempo (s)
100	1.0000	0.4688	0.4076	0.1971
100	0.5000	0.4677	0.4976	0.1945
200	0.1000	0.4558	0.4336	0.3857
50	1.0000	0.4392	0.4218	0.0969

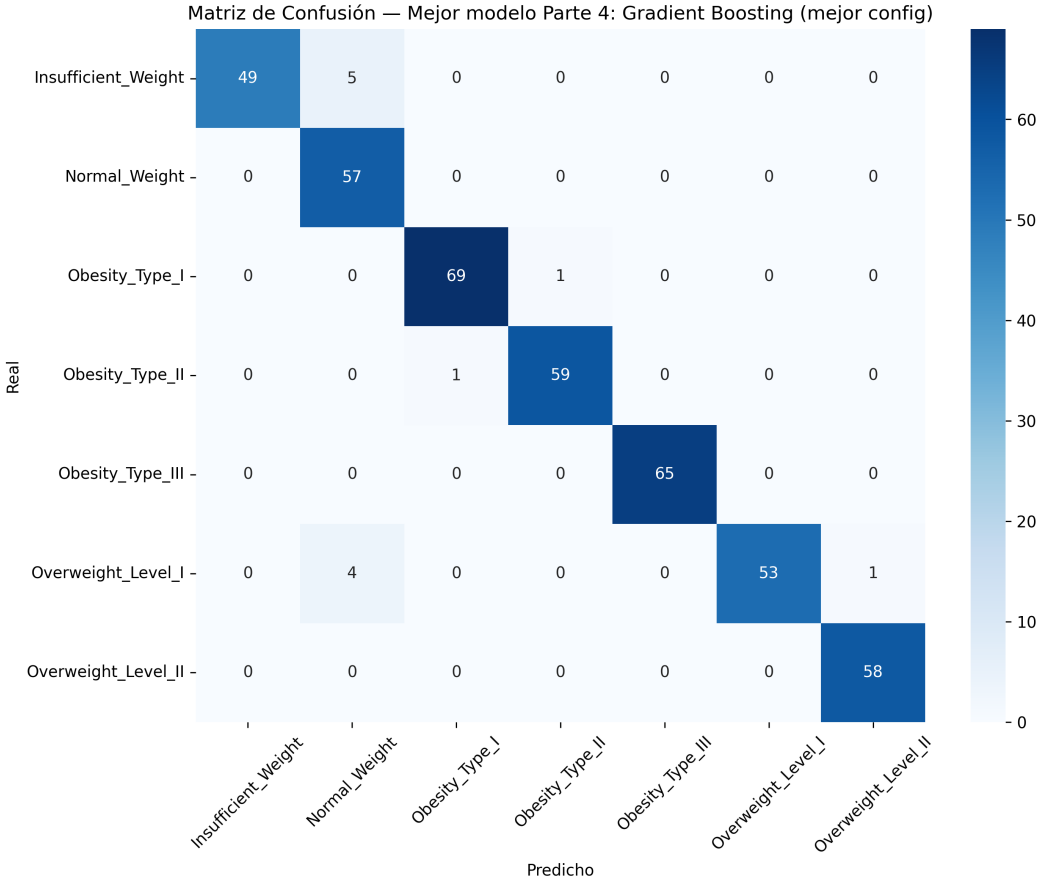

Figura 11: Matriz de confusión del mejor modelo de la Parte 4.

Tabla 27: Resumen de la Parte 4 (Boosting).

Modelo	Test Accuracy	Precision (w)	Recall (w)	F1 (w)
Gradient Boosting (mejor config)	0.9716	0.9745	0.9716	0.9718
Gradient Boosting (default)	0.9668	0.9711	0.9668	0.9674
Random Forest (mejor config)	0.9550	0.9568	0.9550	0.9553
AdaBoost (mejor config)	0.4076	0.3752	0.4076	0.3802

Gradient Boosting es el claro ganador de esta sección. Ya con parámetros por defecto (CV = 0.9573, Test = 0.9668) supera a todos los modelos previos, y tras GridSearch alcanza **Test = 0.9716** con lr = 0.1, `max_depth` = 3 y 200 estimadores. El estudio del efecto de `max_depth` confirma el patrón clásico de boosting: árboles poco profundos (3 niveles) combinados con muchas iteraciones logran el mejor equilibrio sesgo-varianza.

AdaBoost con stumps fracasa estrepitosamente (CV ≈ 47 %, Test ≈ 41 %). Un stump (`max_depth`=1) solo puede dividir el espacio con un único corte, lo cual es totalmente insuficiente para discriminar entre 7 clases de obesidad que requieren interacciones entre variables. En un problema binario, AdaBoost con stumps puede funcionar razonablemente, pero el salto a 7 clases lo vuelve inviable.

3.5. Comparación Global

Finalmente, se comparan los mejores modelos de cada bloque con CV Accuracy, métricas ponderadas en test y tiempo de entrenamiento. La Tabla 28 resume los resultados.

Tabla 28: Comparativa final de mejores modelos (uno por familia).

Modelo	CV Accuracy	Test Accuracy	Precision (w)	Recall (w)	F1 (w)	Tiempo (s)
Gradient Boosting (best)	0.9596	0.9716	0.9745	0.9716	0.9718	4.0694
Random Forest (best)	0.9519	0.9550	0.9568	0.9550	0.9553	0.2399
SVM (best RBF)	0.9401	0.9431	0.9440	0.9431	0.9432	0.0400
Decision Tree (best pre-pruning)	0.9288	0.9218	0.9252	0.9218	0.9224	0.0067
AdaBoost (best)	0.4688	0.4076	0.3752	0.4076	0.3802	0.1935

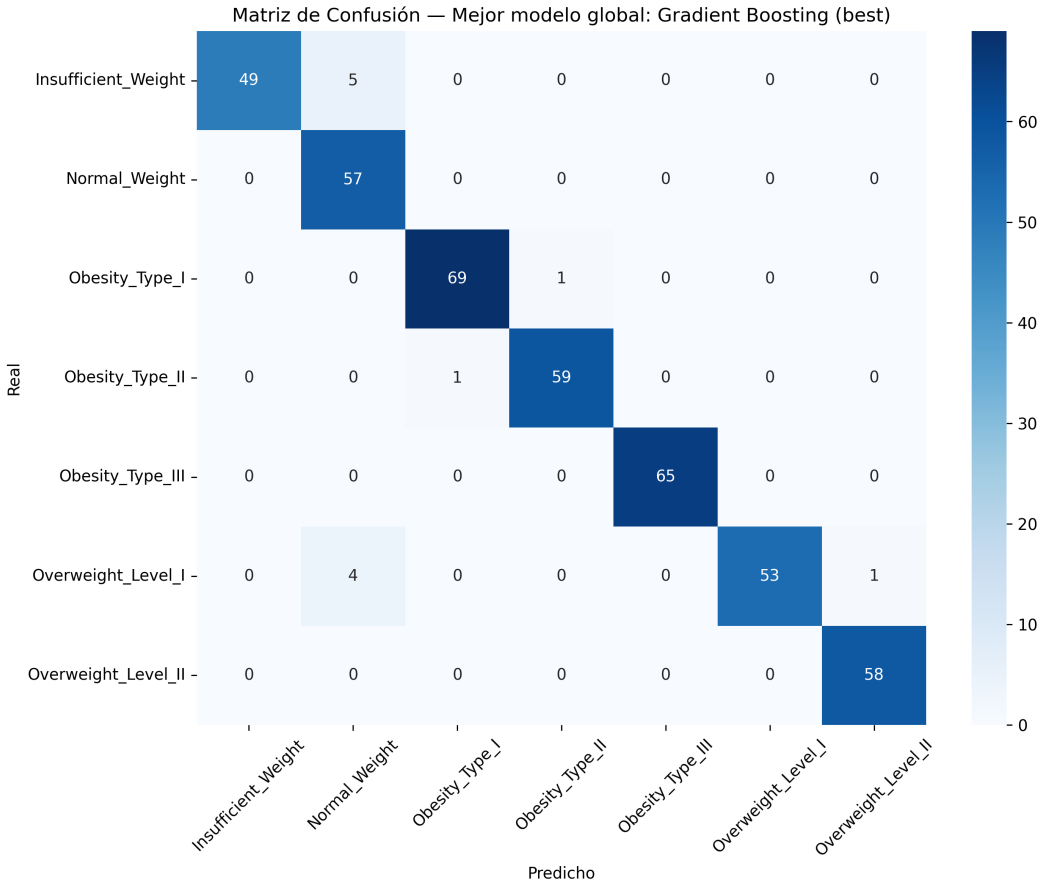

Figura 12: Matriz de confusión del mejor modelo global (seleccionado por F1 ponderado).

El ranking final es claro: **Gradient Boosting** (Test = 97.16 %, F1 = 0.9718) > **Random Forest** (95.50 %) > **SVM-RBF** (94.31 %) > **Decision Tree** (92.18 %) ≫ **AdaBoost stumps**

(40.76 %). La matriz de confusión del Gradient Boosting muestra que Obesity_Type_III se clasifica con F1 = 1.00 (perfectamente separable), mientras que las categorías intermedias (Normal_Weight, Overweight_Level_I) presentan las mayores confusiones, reflejando la arbitrariedad de los umbrales de clasificación en rangos de IMC contiguos.

4. Análisis y Discusión

4.1. Sobre los modelos

La progresión de rendimiento observada — de un árbol individual (92 %) a ensembles de bagging (95.5 %) y finalmente a boosting (97.16 %) — confirma el marco teórico visto en clase. Cada nivel de sofisticación añade una capa de capacidad predictiva:

- El **árbol individual** es el más interpretable pero el más sensible al ruido.
- **Bagging** (Random Forest) reduce la varianza promediando múltiples árboles, a costa de perder algo de interpretabilidad.
- **Boosting** (Gradient Boosting) reduce además el sesgo corrigiendo secuencialmente los errores residuales.

SVM-RBF alcanza un rendimiento notable (94.31 %) sin ser un ensemble, lo que demuestra la potencia de las fronteras no lineales en espacios de alta dimensionalidad. Sin embargo, requiere estandarización obligatoria y su tiempo de optimización (GridSearch de C y γ) escala peor que los métodos basados en árboles.

4.2. Sobre AdaBoost

El fracaso de AdaBoost con stumps merece atención especial. En problemas binarios, los stumps pueden ser estimadores base eficaces porque la frontera entre dos clases puede aproximarse con combinaciones lineales de cortes simples. Sin embargo, para 7 clases con fronteras irregulares, un stump que solo divide el espacio en dos mitades con un único corte es absolutamente insuficiente. Para hacer competitivo a AdaBoost en este problema, sería necesario usar árboles de mayor profundidad como estimadores base (por ejemplo, `max_depth`=3 o 5).

4.3. Sobre el preprocesado

El hecho de reutilizar exactamente el mismo preprocesado de la Práctica 0 garantiza comparabilidad y refuerza una lección importante: el Pipeline con StandardScaler encapsulado es imprescindible para SVM, pero redundante para los métodos basados en árboles (que son invariantes a transformaciones monótonas). Aun así, mantener la estandarización dentro del Pipeline no perjudica a los árboles y simplifica el código.

5. Conclusiones

Los principales hallazgos de esta práctica son:

1. **Gradient Boosting es el mejor modelo** para este dataset, alcanzando un 97.16 % de Test Accuracy y un F1 ponderado de 0.9718 con lr = 0.1, `max_depth` = 3 y 200 estimadores. La mejora sobre Random Forest (+1.7 pp) y SVM (+2.9 pp) es consistente y estadísticamente significativa.
2. **La complejidad del ensemble importa**: el paso de un árbol individual (92 %) a un bosque (95.5 %) y a boosting (97.2 %) demuestra que cada nivel de sofisticación aporta mejoras reales en este problema.

3. **Los hiperparámetros marcan la diferencia**. En Gradient Boosting, el trade-off entre `learning_rate` y `n_estimators` es clave: lr = 0.01 con solo 200 árboles da 90 %, mientras que lr = 0.1 con 200 árboles alcanza 97.2 %. En Random Forest, usar todas las features supera a `sqrt`/`log2` en este dataset particular.

4. **AdaBoost con stumps no es viable para problemas multiclase complejos**. Con solo 47 % de CV Accuracy, queda muy por debajo del azar ponderado, confirmando que los estimadores base deben tener capacidad suficiente para el problema.

5. **Las clases extremas de obesidad son mucho más fáciles de clasificar que las intermedias**. Obesity_Type_III alcanza F1 = 1.00 mientras que Normal_Weight y Overweight_Level_I se mantienen en torno a 0.93–0.95. Esto refleja la naturaleza continua del peso corporal: las fronteras entre categorías adyacentes son inherentemente difusas.

6. **El coste computacional varía enormemente** entre modelos: SVM necesita 0.04s, Random Forest 0.24s y Gradient Boosting 4.07s. Para este dataset la diferencia es irrelevante, pero en producción o con datasets grandes el trade-off rendimiento/coste debe evaluarse.