Máster en Ingeniería Informática

Análisis Inteligente de Datos 24/25

Grupo 1

Proyecto Final

«Aplicación de *Machine Learning*»

Luis Daniel Casais Mezquida — 100429021

Profesor

Pablo Gutierrez Ruiz

Tabla de Contenidos

Lista de Figuras

1. Análisis exploratorio de datos

1.1. Elección del *dataset*

Para la realización de la práctica, se eligió el *dataset* "Obesity Prediction Dataset"[1], el cual contiene tiene como objetico estimar los los niveles de obesidad de individuos de México, Perú, y Colombia, dada información sobre sus hábitos alimenticios y físicos. Cuenta con 2111 registros.

El *dataset* consta de las siguientes variables:

- *Gender*: Género (sexo) de la persona
- *Age*: Edad de la persona
- *Height*: Altura, en metros
- *family_history_with_overweight*: Si la persona tiene un historial familiar de sobrepeso
- *FAVC*: Si la persona consume alimentos con alto contenido calórico
- *FCVC*: Frecuencia de consumición de vegetales (escala de 1 a 3).
- *NCP*: Número de comidas principales diarias
- *CAEC*: Frecuencia de consumición de alimentos entre comidas (*no*, *Sometimes*, *Frequently*, *Always*)
- *SMOKE*: Si la persona fuma
- *CH2O*: Consumición diaria de agua (escala de 1 a 3)
- *SCC*: Si la persona monitoriza las calorías que consume
- *FAF*: Frecuencia de actividad física (escala de 0 a 3)
- *TUE*: Tiempo gastado usando dispositivos electrónicos (escala de 0 a 3)
- *CALC*: Frecuendia de consumición de alcohol (*no*, *Sometimes*, *Frequently*, *Always*)
- *MTRANS*: Principal modo de transporte (*Automobile*, *Bike*, *Motorbike*, *Public Transportation*, *Walking*)
- *NObeyesdad*: Nivel de obesidad (*Insufficient Weight*, *Normal Weight*, *Overweight Level I*, *Overweight Level II*, *Obesity Type I*, *Obesity Type II*, *Obesity Type III*).

1.2. Estadísticas descriptivas

Empezaremos con un análisis del *dataset* elegido, para obtener posible información oculta en el mismo. Para éste análisis, se ha usado la librería de Python `ydata-profiling`[2].

Las principales características de los datos quedan recogidas en la Tabla 1 y Tabla 2.

[1] https://www.kaggle.com/datasets/adeniranstephen/obesity-prediction-dataset
[2] https://github.com/ydataai/ydata-profiling

Tabla 1: Estadísticas del *dataset*.

Número de características	17
Número de observaciones	2112
Celdas vacías	0
Registros duplicados	9 (33)
% duplicados	1.56%

Tabla 2: Tipos de características.

Categóricas	5
Numéricas	8
Booleanas	4

Como podemos observar, no hay celdas vacías y los tipos de variables usadas están perfectamente descritas y enumerados, por lo que es fácil tratar con los datos. Sin embargo, también podemos observar que hay un número significativo de registros duplicados.

Analizando estos casos en la Figura 1, vemos que la mayoría están duplicadas de dos a cuatro veces, mientras que hay una en específico que está duplicada catorce veces.

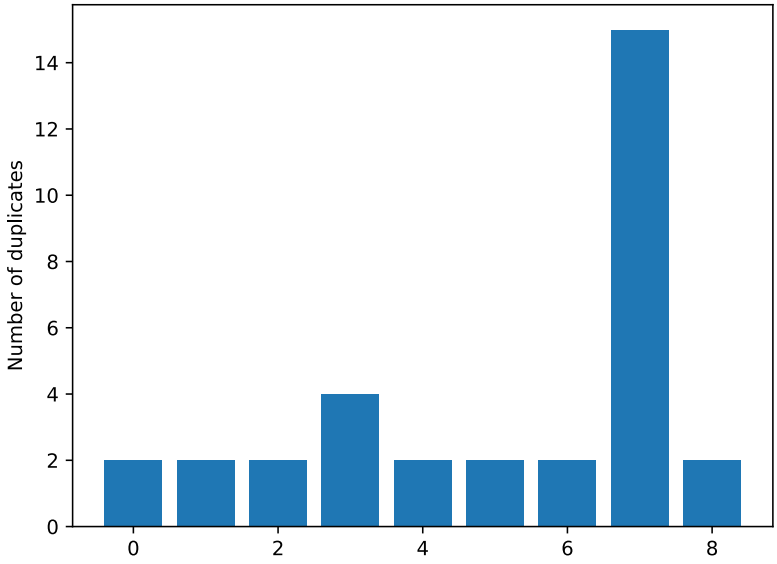


Figura 1: Cantidad de registros duplicados por registro duplicado

En la Tabla 3 se puede observar con más detenimiento éste caso. Dado que estos duplicados suponen apenas un 0.71% del *dataset*, y dado que se refieren a varones de edad joven y estatura y peso medio para su país de origen, podríamos asumir que se trata de una simple casualidad.

Tabla 3: Registro con mayor número de duplicados

Gender	Male
Age	21
Height	1.62
Weight	70.0
family_history_with_overweight	no
FAVC	yes
FCVC	2.0
NCP	1.0
CAEC	no
SMOKE	no
CH2O	3.0
SCC	no
FAF	1.0
TUE	0.0
CALC	Sometimes
MTRANS	Public_Transportation
NObeyesdad	Overweight_Level_I
# duplicates	15

1.3. Evaluación de las características

A continuación evaluaremos las características para observar qué tipo de preprocesado es necesario realizar para mejorar el entrenamiento.

1.3.1. Análisis univariable

Primero, analizaremos las distintas características de forma aislada, y mencionaré las más relevantes[3].

Hay características muy desbalanceadas, como son *family_history_with_overweight* (*True*, 81.8%), *FAVC* (*True*, 88.4%), *CAEC* (*Sometimes*, 83.6%), *SMOKE* (valor *False*, 97.9%), *SCC* (*False*, 95.5%), *MTRANS* (*Public_Transportation*, 74.8%). Sin embargo, la característica objetivo, *NObeyesdad*, está relativamente balanceada, como se puede observar en la Figura 2.

Con respecto a las variables de rangos numéricos (*FCVC*, *NCP*, *CH2O*, *FAF*, *TUE*), podemos observar que un pequeño porcentaje de los registros contienen valores entre los rangos discretos, lo cual puede ser debido a errores en la recogida de los datos. Un ejemplo de esto queda reflejado en la Figura 3.

[3]Para un resumen detallado, ver el *notebook* `analysis.ipynb`.

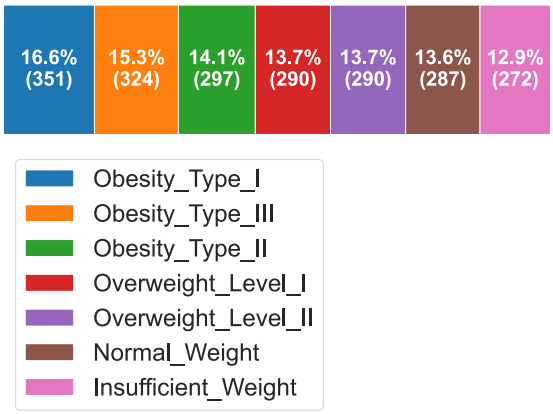


Figura 2: Distribución de la característica *NObeyesdad*.

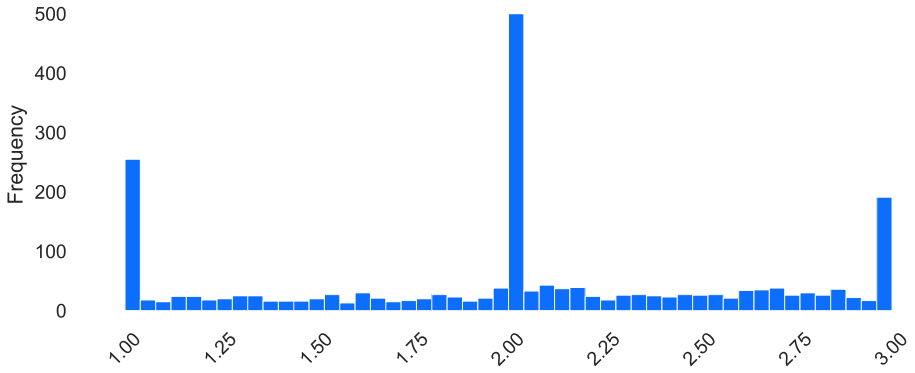


Figura 3: Distribución de la característica *CH2O*.

1.3.2. Análisis multivariable

La Figura 4 muestra la correlación entre las distintas características. Si nos fijamos en la columna de *NObeyesdad*, observamos que tanto el género, como el peso y el historial familiar con el peso están bastante relacionados con la obesidad. Los dos primeros tienen sentido, pero es el caso del género el que resulta extraño. Si observamos más detenidamente el tipo de obesidad por género (véase Figura 5), podemos ver que en el caso de Obesidad de tipo II, no hay apenas mujeres, mientras que en el caso del tipo III, no hay apenas hombres. Sin embargo, la característica está muy balanceada lo cual, al ser nuestro *target*, es algo muy beneficioso, porque implica que nuestras clases están balanceadas.

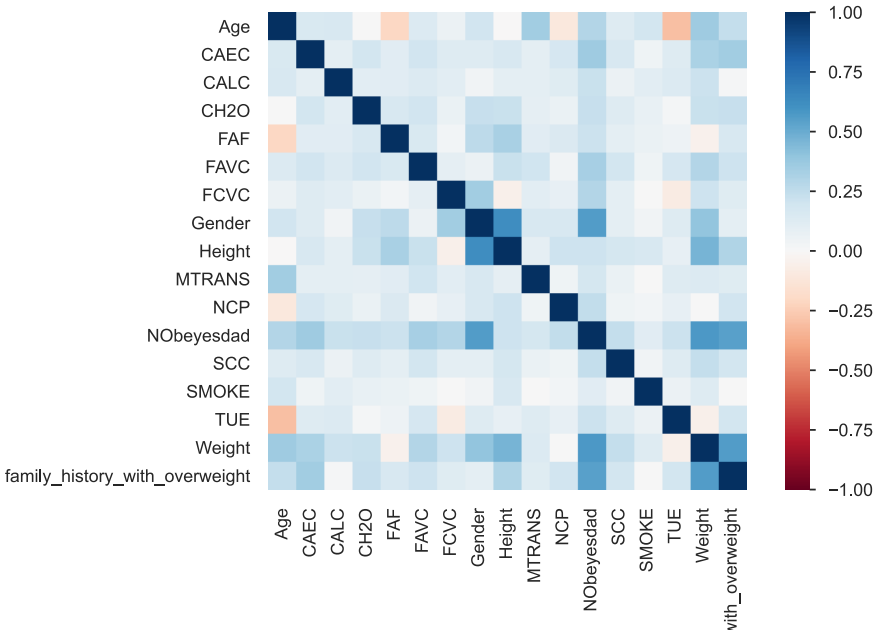


Figura 4: Correlación entre las distintas características.

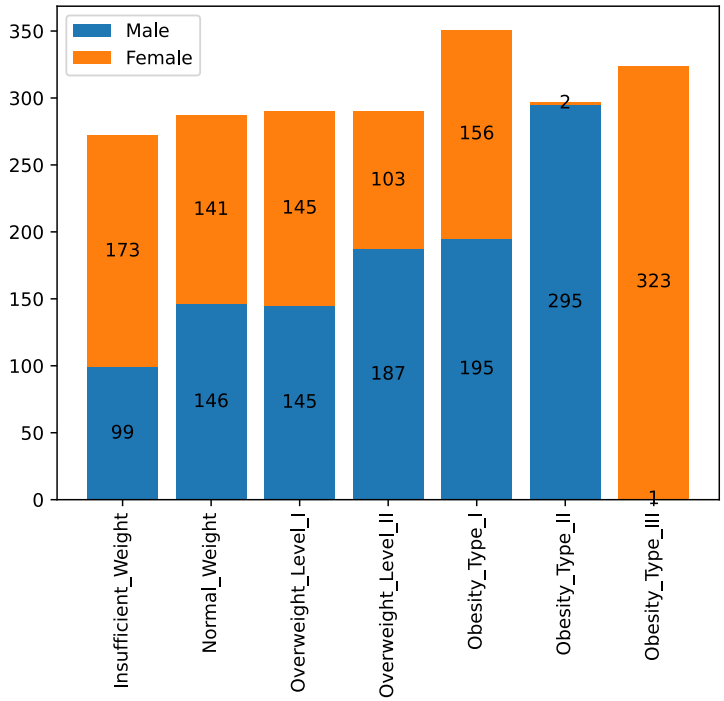


Figura 5: Correlación entre el género y el tipo de obesidad.

2. Metodología

2.1. Elección de la métrica de evaluación

El problema que nos encontramos es uno de clasificación multiclase. Para asegurarnos de que el clasificador es versátil y damos la misma importancia a la precisión y la exhaustividad, se decidió usar como métrica de evaluación el *F1-score*, específicamente en su versión *macro*, dado que, como mencionamos en la Figura 2, el *target* está relativamente balanceado.

2.2. Preparación del *dataset*

Como observamos en la Sección 1.3.1, el *dataset* está formado por variables tanto binarias, de rangos numéricos, y enumeradas. Para las variables binarias y enumeradas, es necesario codificarlas en valores numéricos para el entrenamiento, ya que las entradas de nuestros modelos son numéricas. Para las variables numéricas, se aplicó escalado[4].

También se observó que las variables de rango contaban con un pequeño porcentaje de valores no enteros, los cuales se decidieron redondear al entero más cercano, ya que asumimos que pueden tratarse de errores en el *dataset*, ya que en esta característica no vemos necesario el uso de valores tan precisos.

[4] Para observar la codificación en detalle, ver el *notebook* `models.ipynb`.

3. Experimentación

Para el proceso de experimentación, se empezará realizando pruebas con distintos modelos para ver sus resultados (Sección 3.1), y luego se probará a combinar los mejores modelos mediante la técnica de *stacking* para observar si se obtiene un mejor resultado (Sección 3.2).

Para todos los entrenamientos, se dividió el *dataset* en 80% datos de entrenamiento y 20% datos de prueba, y se usó un valor de *Cross Validation* de 5.

3.1. Modelos base

Para la experimentación con modelos base, se decidió probar con modelos más sencillos y más avanzados, ya que no siempre un modelo más avanzado implica un resultado mejor.

Dado que se trata de un problema de clasificación multiclase, se eligieron los siguientes modelos[5]:

1. *Naive Bayes*
2. *K-Neighbours*
3. *SVM*
4. *Logistic Regression*
5. *Random Forest*
6. *XGBoost*

Para cada uno de los modelos (exceptuando *Naive Bayes*), se procedió también a un ajuste de los hiperparámetros más prevalentes, para así también obtener las mejores versiones de cada uno de los modelos para el problema.

También se guardó información del tiempo que duró el entrenamiento de cada uno de los modelos, aúnque este incluye la búsqueda de los hiperparámetros óptimos.

3.1.1. Resultados

La Figura 6 muestra los resultados de cada uno de los modelos utilizados.

Se puede observar que el modelo más simple, *Naive Bayes*, tiene un resultado verdaderamente horroroso, con f1_macro < 0.5, el cual indica que no ha aprendido nada. Al ser un problema multiclase y no binario, no podemos simplemente invertir la salida del clasificador para obtener un resultado mejor.

El modelo *K-Neighbours* obtuvo un resultado más que aceptable, aún siendo un modelo simple, probablemente debido a la alta correlación de algunas de las características con la característica objetiva. Los mejores hiperparámetros encontrados fueron:

- Métrica: Distancia *Manhattan*
- Número de vecinos: 5
- Función de pesos: Inverso de la distancia

[5] Para más información sobre los modelos exactos utilizados, ver el *notebook* `models.ipynb`.

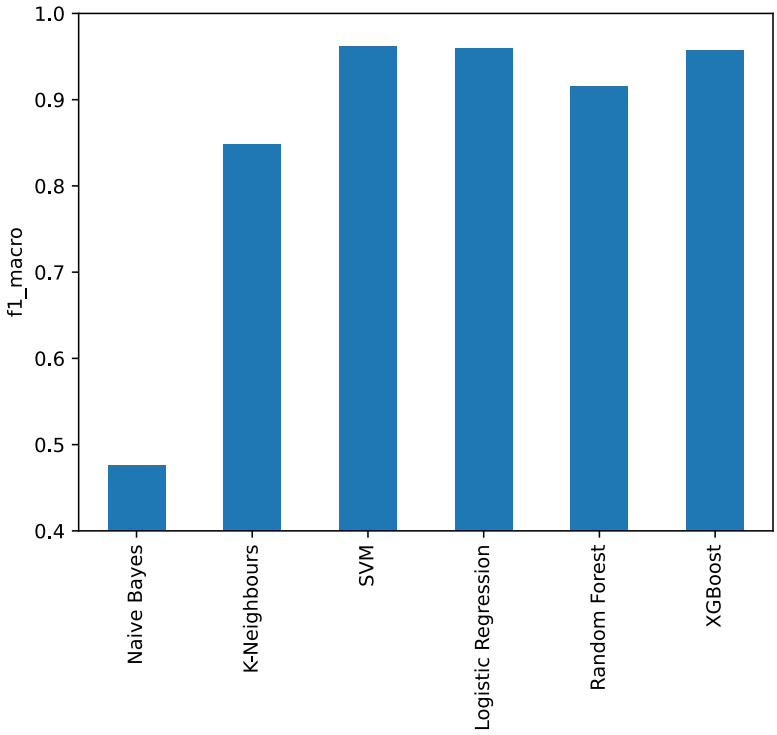


Figura 6: Resultado de los distintos modelos empleados.

El modelo *SVM* obtuvo un resultado excelente y, dado que el conjunto de datos es pequeño, lo convierte en un buen candidato. Los mejores hiperparámetros encontrados fueron:

- C: 10
- *Kernel*: Lineal

El modelo *Logistic Regression*, obtuvo unos resultados también excelentes. Cabe destacar que en algunas pruebas experimentales, éste modelo fue capar de obtener resultados con f1_macro > 0.9 incluso con un 30% del *dataset* total dedicado al entrenamiento, y el resto a la prueba, por lo que no está haciendo *overfitting*. Los mejores hiperparámetros encontrados fueron:

- C: 10
- Penalización: L1
- *Solver*: SAGA

El modelo *Random Forest* obtuvo unos resultados igualmente excelentes, pero ligeramente inferiores a sus compañeros. Los mejores hiperparámetros encontrados fueron:

- Profundidad máxima: 20
- Mínimo número de muestras para el *split*: 2
- Número de estimadores: 100

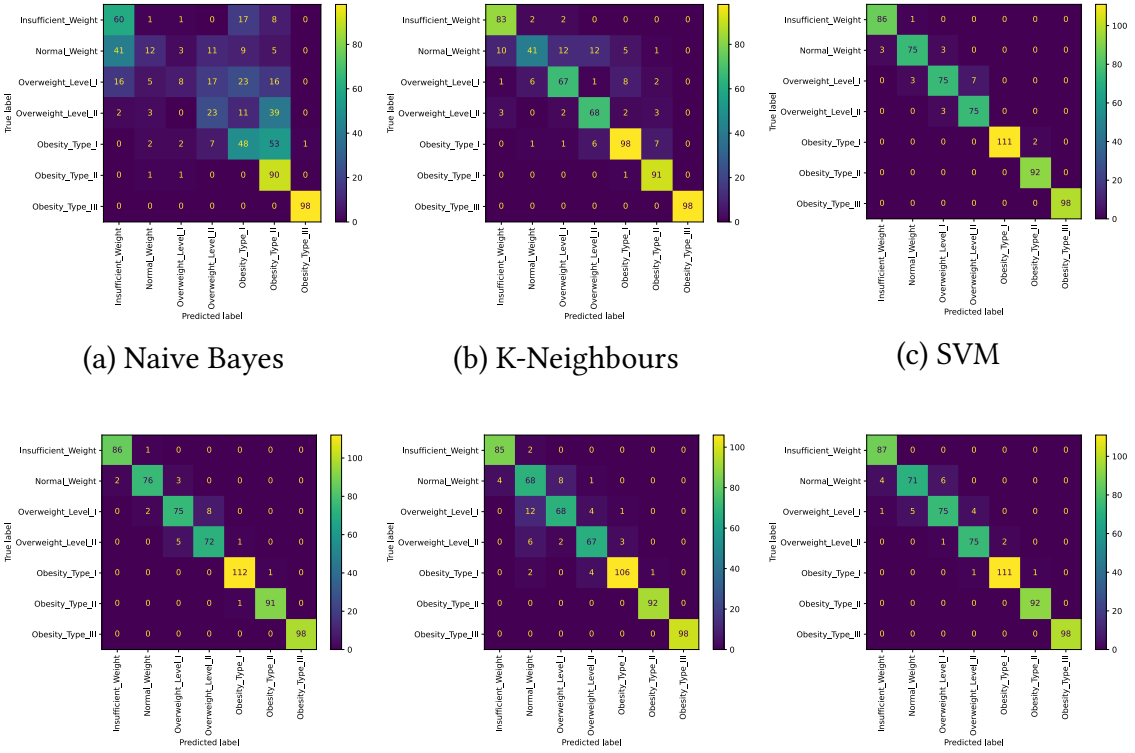
Finalmente, se probó un tipo de modelo más avanzado, *XGBoost*, el cual también obtuvo unos resultados excelentes, pero ligeramente por debajo de *SVM* y *Logistic Regression*. Los mejores hiperparámetros encontrados fueron:

- Tasa de aprendizaje: 0.2

- Profundidad máxima: 3
- Número de estimadores: 200

La Figura 7 muestra las matrices de confusión de los distintos modelos, las cuales muestran que, en general, se suelen confundir más los tipos de peso normales y con sobrepeso leve.

Finalmente, la Figura 8 muestra el tiempo empleado en el entrenamiento de cada uno de los modelos. Se observa que, de entre los mejores modelos, *SVM* cuenta con el entenamiento más veloz, aunque es posible que sea debido en parte al menor número de hiperparámetros buscados.



(a) Naive Bayes (b) K-Neighbours (c) SVM

(d) Logistic Regression (e) Random Forest (f) XGBoost

Figura 7: Matrices de confusión de los modelos.

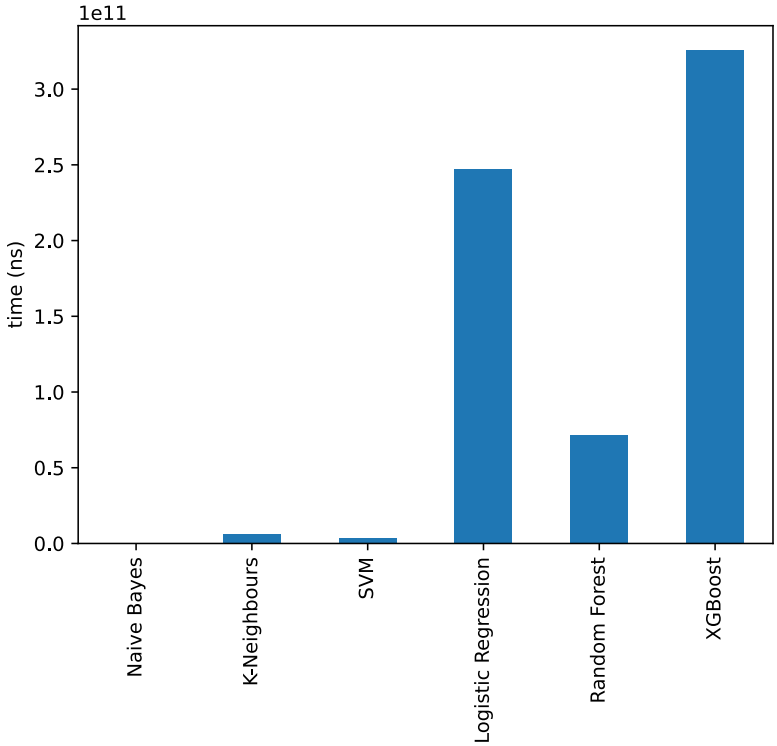


Figura 8: Tiempo de entrenamiento de los distintos modelos empleados.

3.2. *Stacking*

Como último paso, se decidió aplicar la técnica de *stacking* con el objetivo de comprobar si producía mejores resultados que los modelos base. Para ello, se decidió usar los tres mejores modelos, *SVM*, *Logistic Regression*, y *XGBoost*, con los mejores hiperparámetros encontrados. Para el estimador final, se usó otro modelo *Logistic Regression*.

3.2.1. Resultados

La Figura 9 muestra los resultados de cada uno de los modelos utilizados. Se puede observar que el uso de ésta técnica ha resultado muy útil, obteniendo un resultado de f1_macro ≈ 0.97.

La Figura 10 muestra la matriz de confusión, la cual es similar a las del resto de modelos usados. La Figura 11 muestra el tiempo de entrenamiento comparado con el tiempo del resto de modelos usados. Al no tener que buscar hiperparámetros, el tiempo es considerablemente inferior.

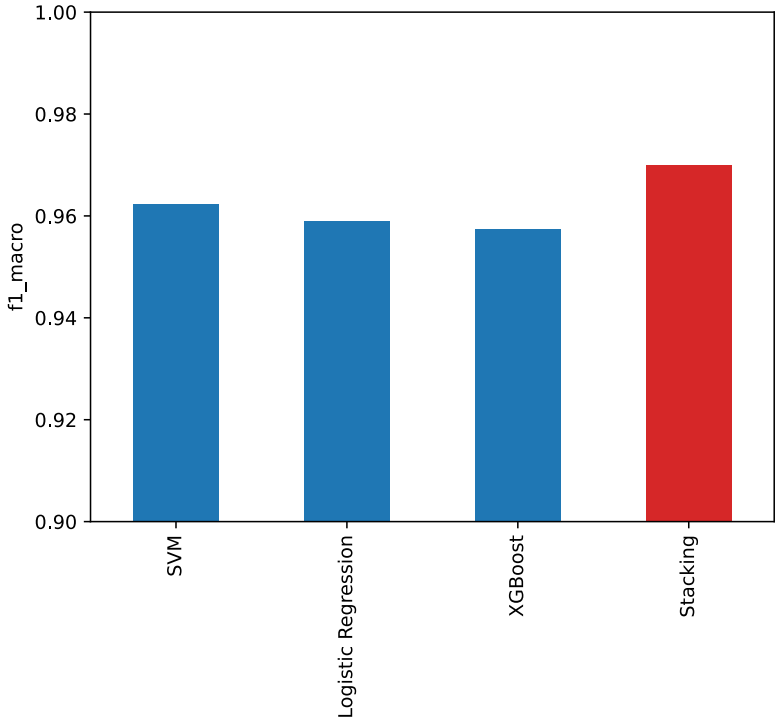


Figura 9: Resultado de la técnica de *stacking*.

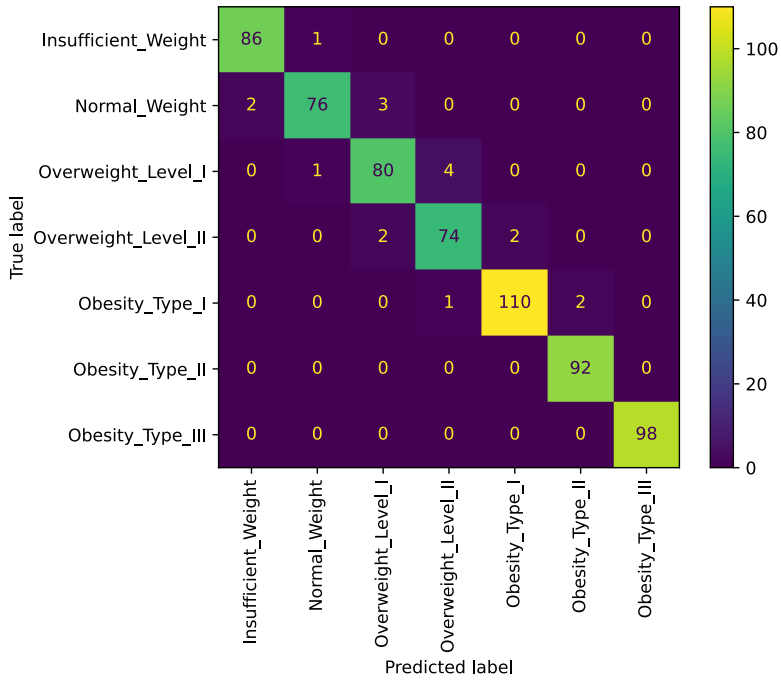


Figura 10: Matriz de confusión de la técnica de *stacking*.

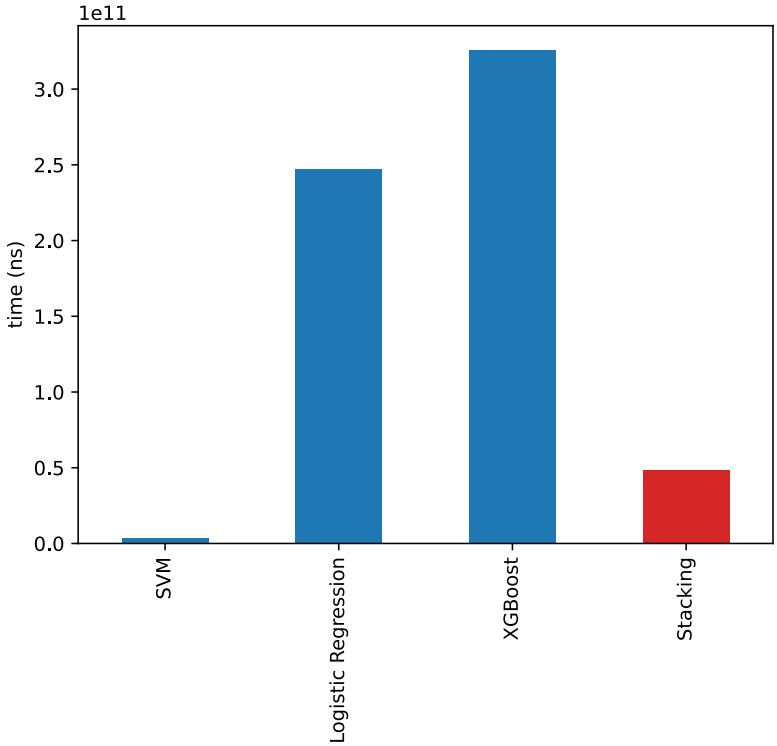


Figura 11: Tiempo de entrenamiento de la técnica de *stacking*.

4. Modelo elegido

Tras analizar los resultados de los distintos modelos, se ha decidido elegir el modelo de *stacking*, tal y como ha sido descrito en la Sección 3.2, debido a su altísimo resultado en la métrica seleccionada, y con un coste en tiempo de entrenamiento aceptable.

Sin embargo, cabe realizar una mención especial al modelo *SVM*, el cual obtuvo uno de los mejores resultados con un muy bajo tiempo de entrenamiento, aunque en el caso de tratar con *datasets* más grandes, éste se podría disparar.