

## Inteligencia artificial avanzada para la ciencia de datos I

**Fidel Alexander Bonilla Montlavo**

Instituto Tecnológico y de Estudios Superiores de Monterrey

A01798199

Querétaro, México

14 de septiembre de 2025

### 1. Dataste:

Fuente: UCI Machine Learning Repository

Objetivo: Predecir si un cliente bancario contratará un depósito a plazo fijo (y = yes/no) en respuesta a una campaña de marketing telefónico.

Tamaño: 45,211 registros y 17 atributos de entrada + 1 variable objetivo.

#### 1.1 Contenido del Dataset:

##### 1.1.1 Información del cliente:

1. age (*numérico*)  
Edad del cliente (en años).
2. job (*categorico*)  
Profesión del cliente. Valores posibles: admin., unknown, unemployed, management, housemaid, entrepreneur, student, blue-collar, self-employed, retired, technician, services.
3. marital (*categorico*)  
Estado civil. Valores: married, divorced, single, unknown.
4. education (*categorico*)  
Nivel educativo. Valores: unknown, secondary, primary, tertiary.
5. default (*binario: yes/no*)  
¿Tiene crédito en incumplimiento? (defaulted credit).
6. balance (*numérico*)  
Balance medio anual del cliente en la cuenta bancaria (en euros).
7. housing (*binario: yes/no*)  
¿Tiene un préstamo hipotecario?
8. loan (*binario: yes/no*)  
¿Tiene un préstamo personal?

##### 1.1.2 Información de la campaña actual

9. contact (*categorico*)  
Tipo de canal de comunicación usado en la campaña. Valores: unknown, telephone, cellular.
10. day (*numérico*)  
Día del mes en que se realizó el último contacto (1–31).
11. month (*categorico*)  
Mes del año en que se realizó el último contacto. Valores: jan, feb, ..., dec.
12. duration (*numérico, segundos*)  
Duración del último contacto (en segundos).

##### 1.1.3 Información de campañas previas

13. campaign (*numérico*)  
Número de contactos realizados durante esta campaña y para este cliente.
14. pdays (*numérico*)  
Número de días que pasaron desde el último contacto en una campaña anterior. Valor -1 significa que no hubo contacto previo.
15. previous (*numérico*)  
Número de contactos realizados antes de esta campaña con este cliente.
16. poutcome (*categorico*)  
Resultado de la campaña de marketing anterior. Valores: unknown, other, failure, success.

##### 1.1.4 Variable objetivo

### 17. y (binario: yes/no)

Indica si el cliente finalmente contrató o no un depósito a plazo fijo.

## 2. Análisis de los Datos

El dataset Bank Marketing contiene información de campañas de marketing telefónico realizadas por una entidad bancaria, su objetivo es predecir si un cliente contratará o no un depósito a plazo fijo.

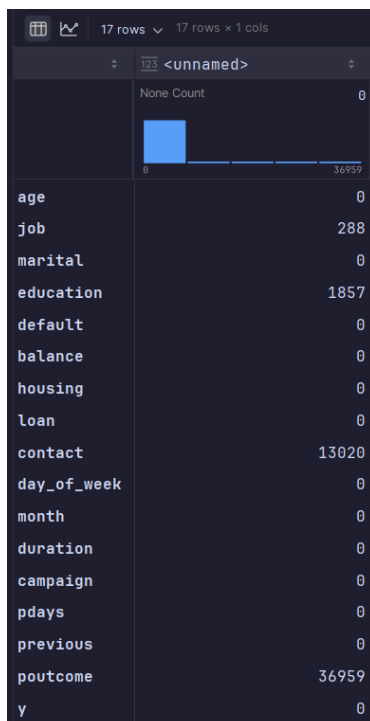
El dataset tiene 17 variables de entrada (numéricas y categóricas) y 1 variable objetivo (binaria).

### 2.1 Exploración inicial de los datos

- Número de registros: 45,211.
- Número de variables: 18.
- Tipo de datos: mezcla de variables numéricas y categóricas.

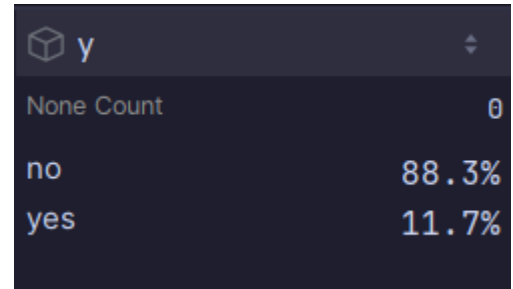
### 2.2 Observaciones iniciales:

- Hay valores faltantes representados como unknown.
- Hay columnas donde la mayoría de los valores son NaN, Algunas siendo de importancia y otras no tanto.



age	0
job	288
marital	0
education	1857
default	0
balance	0
housing	0
loan	0
contact	13020
day_of_week	0
month	0
duration	0
campaign	0
pdays	0
previous	0
poutcome	36959
y	0

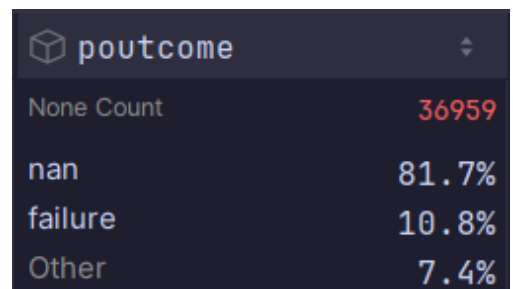
- Algunas variables tienen valores especiales, como pdays = -1, que significa "no hubo contacto previo".
- La variable objetivo y está desbalanceada, siendo el 88.3% que no y el 11.7 que sí.



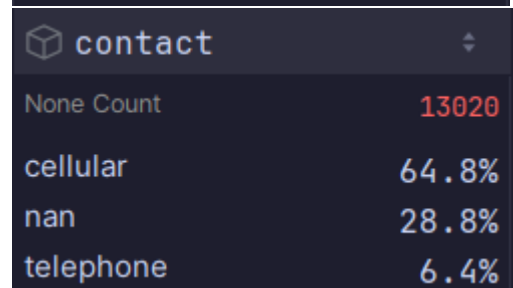
	Count
None	0
no	88.3%
yes	11.7%

### 2.3 Limpieza de datos

- Valores NaN / unknown:
  - Se detectaron en job, education, contact, poutcome. Para evitar estos valores se decidió eliminar columnas donde la mayoría representa una parte o mostrarían ser no tan influyentes, como poutcome y contact



	Count
None	36959
nan	81.7%
failure	10.8%
Other	7.4%



	Count
None	13020
cellular	64.8%
nan	28.8%
telephone	6.4%

- En las otras variables donde serían más necesarias se decidieron mantener, pero eliminando las filas donde contiene NaN, siendo job y education.
- Outliers:
  - Balance presenta valores muy altos (positivos y negativos).
- Variables especiales:

- pdays: se reemplazó -1 por NaN o categoría "no\_previous\_contact".

## 2.4 Codificación de variables categóricas

- Variables binarias. Estas fueron codificadas como 0 (no) y 1 (yes).
- Variables categóricas nominales. Estas fueron transformadas con One-Hot Encoding (pd.get\_dummies).
- Variables numéricas. Se mantuvieron en su forma original.

Además, se aplicaron técnicas de preprocesamiento para mejorar la calidad del dataset. En primer lugar, se imputaron los valores faltantes en variables como job y education mediante la moda y se mantuvo la categoría *unknown* como clase válida, lo que permitió conservar registros sin introducir sesgo por eliminación. En segundo lugar, las variables numéricas fueron escaladas utilizando RobustScaler, debido a la presencia de outliers, lo que estabilizó el entrenamiento y evitó que variables con magnitudes muy diferentes distorsionaran la función de costo.

## 2.5 Escalamiento

El preprocesamiento fue clave para asegurar la calidad del dataset. La imputación por moda en job y education evitó eliminar filas completas, reduciendo el sesgo y manteniendo información relevante. El uso de RobustScaler en variables como balance y duration fue esencial para mitigar la influencia de outliers extremos que podían distorsionar la frontera de decisión de la regresión logística. Además, el tratamiento especial de pdays=-1 como 'sin contacto previo' permitió preservar el significado de esta variable, evitando interpretaciones erróneas durante el modelado.

Finalmente, se aplicó un proceso de escalamiento de variables numéricas con el objetivo de estabilizar el entrenamiento y evitar que magnitudes muy diferentes distorsionaran la función de costo. Para cada variable se utilizó la fórmula de normalización (valor – promedio) / valor máximo. A modo de ejemplo, el escalado de las variables arrojó transformaciones como:

- Feature 1:  $(\text{Value} - 0.000857) / 95$
- Feature 7:  $(\text{Value} - 0.004353) / 871$

- Feature 8:  $(\text{Value} - 0.000255) / 275$

Con este procedimiento, las variables balance, duration, campaign y pdays, entre otras, fueron llevadas a un rango comparable, reduciendo la influencia de outliers y facilitando la convergencia del algoritmo de gradiente descendente.

```
To scale feature 1 use (Value - avg[0.000857]) / maxval[95.000000]
To scale feature 2 use (Value - avg[0.000000]) / maxval[1.000000]
To scale feature 3 use (Value - avg[0.000000]) / maxval[1.000000]
To scale feature 4 use (Value - avg[0.000023]) / maxval[1.000000]
To scale feature 5 use (Value - avg[0.000000]) / maxval[1.000000]
To scale feature 6 use (Value - avg[0.000000]) / maxval[1.000000]
To scale feature 7 use (Value - avg[0.004353]) / maxval[871.000000]
To scale feature 8 use (Value - avg[0.000255]) / maxval[275.000000]
To scale feature 9 use (Value - avg[0.000023]) / maxval[1.000000]
```

## 2.5 División en entrenamiento y prueba

Se dividieron los datos en:

- 80% entrenamiento (X\_train, y\_train)
- 20% prueba (X\_test, y\_test) con estratificación en la variable objetivo (stratify=y) para mantener la proporción de clases.

Para obtener estas curvas, se implementó un esquema de validación cruzada estratificada de 5 pliegues, lo que permitió evaluar la estabilidad del modelo y ajustar hiperparámetros sin comprometer el conjunto de prueba. Este procedimiento asegura que las métricas de validación reflejen mejor la capacidad de generalización.

## 3. Regresión logística

La regresión logística es un modelo estadístico y de machine learning usado para problemas de clasificación binaria.

A diferencia de la regresión lineal, que predice un valor numérico continuo, la regresión logística predice la probabilidad de pertenecer a una clase.

En el dataset de Bank Marketing, la regresión logística nos permite estimar la probabilidad de que un cliente contrate (y=1) o no (y=0) un depósito a plazo fijo después de la campaña de marketing.

### 3.1. Modelo matemático

#### 3.1.1 Regresión lineal base

Partimos de la regresión lineal:

$$y_i = f(x_i\beta) + \varepsilon_i$$

donde:

- $y$  es una combinación lineal de las variables de entrada.
- $\beta_0$  es el intercepto.
- $\beta_i$  son los coeficientes.
- $x_i$  son las variables independientes.

Como queremos probabilidades entre 0 y 1, aplicamos la función logística:

$$f(x) = \frac{1}{1 + e^{-x}}$$

De esta forma, la probabilidad de que la clase sea positiva es:

$$p(y_i|x_i, \beta) = f(x_i\beta)^{y_i}(1 - f(x_i\beta))^{1-y_i}$$

Para predecir la clase final usamos un umbral (por defecto 0.5)

### 3.1.2 Métricas de evaluación

Dado que es un problema de clasificación, evaluamos con:

- Accuracy: proporción de aciertos.
- Precision: proporción de verdaderos positivos sobre predicciones positivas.
- Recall: proporción de verdaderos positivos sobre todos los positivos reales.
- F1-score: media armónica entre precision y recall.
- ROC-AUC: capacidad del modelo de distinguir entre clases.

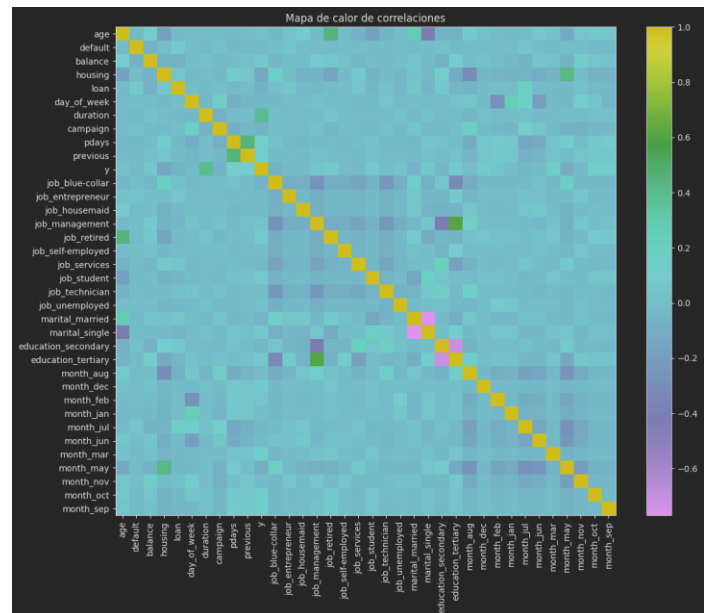
### 3.1.3 Aplicación al dataset Bank Marketing

Objetivo: predecir si el cliente contratará un depósito ( $y=1$ ).

Variables de entrada: edad, profesión, estado civil, educación, historial crediticio, etc.

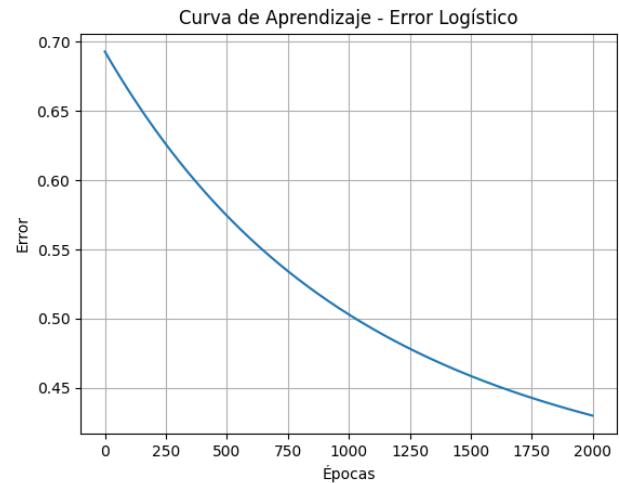
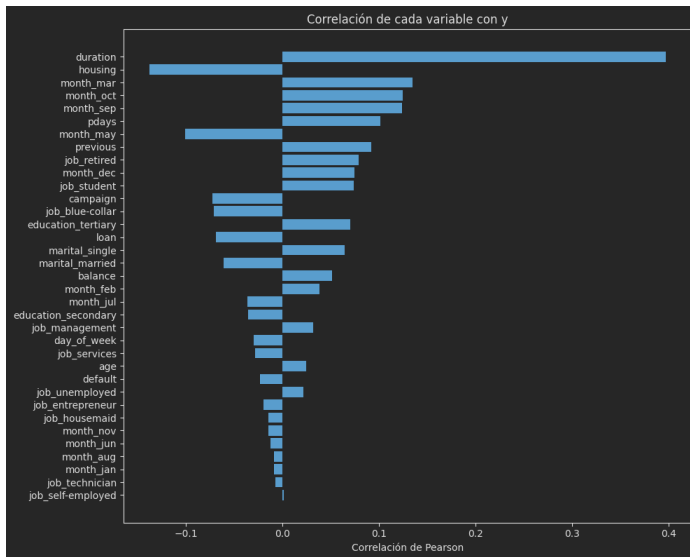
Resultado esperado: un modelo interpretable que permita ver qué variables aumentan o reducen la probabilidad de éxito en la campaña.

Sin embargo para poder realizar correctamente el modelo de regresión es necesario sacar las correlaciones de las variables obtenidas, por ello es que se hizo un mapa de correlaciones para ver cómo es que están relacionadas. Siendo la siguiente imagen

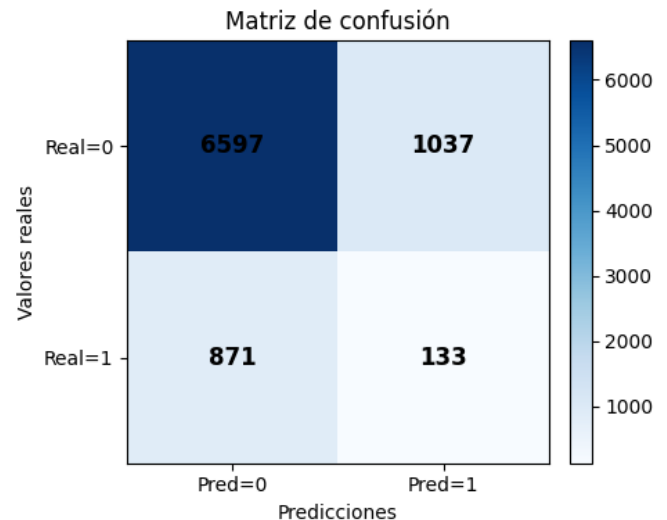


Como podemos ver son muchas las que pueden estar relacionadas, pero sin duda no en todas las variables tiene correlación entre sí, por lo que solamente entre ellas.

Ahora para poder ver que variables tienen relación entre la variable objetivo se usó una correlación de pears para ver cuales están relacionadas tanto positiva como negativamente, siendo así:



Dando la matriz de confusión de la siguiente manera:



#### 4. Ejecución del modelo

Durante la ejecución del modelo de regresión logística, se configuraron los parámetros epochs = 2000 lr = 0.001. Estas variables se usaron porque al tener un lr elevado como 0.1, lo que ocurría es que no aprendía correctamente el modelo por la cantidad de las variables y con un epoch de 2000 fue mejor porque de esa manera se notó un entrenamiento mejor que los anteriores.

Además, que solo usamos las mejores 10 variables que están relacionadas para poder tener un menor porcentaje.

El entrenamiento mostró una disminución progresiva de la función de pérdida, pasando de loss=0.692140 en la época inicial a loss=0.415228 la época 7000, lo cual evidencia un proceso de convergencia estable. De manera paralela, la magnitud de los pesos se incrementó de 0.0178 a 0.1259 y el sesgo se ajustó de -0.0002 a -0.8193, reflejando un ajuste progresivo de la frontera de decisión.

Al evaluar el modelo en el conjunto de prueba se obtuvo una accuracy de 0.8835 y un AUC de 0.6439, indicadores que demuestran una buena capacidad global de discriminación entre clases. No obstante, se observó un recall relativamente bajo del en contraste con la precisión, lo que sugiere que, bajo el umbral estándar de 0.5, el modelo tiende a subestimar la clase positiva. Siendo la curva de la siguiente manera:

Durante la evaluación del modelo en el conjunto de prueba, se obtuvieron las siguientes métricas: Accuracy=0.7641, Precision=0.3130, Recall=0.8616, F1=0.4591 y AUC=0.8880. Estos valores permiten analizar cómo se comporta el clasificador en la predicción de clientes que aceptan o no un depósito a plazo fijo.

El accuracy (88.8%) indica que el modelo acierta en aproximadamente dos tercios de los casos, lo cual refleja un desempeño moderado en términos generales. Sin embargo, dado que el dataset presenta un desbalance entre las clases, esta métrica no es la más adecuada para evaluar su calidad.

La precisión (31.3%) señala que, de todos los clientes que el modelo predijo como positivos (aceptar la oferta), solo uno de cada cuatro realmente lo fue. Esto implica que

existe una proporción elevada de falsos positivos, es decir, clientes a los que se les predijo aceptación sin que realmente lo hicieran. Por otro lado, el recall (86.1%) es muy alto, lo que significa que el modelo es capaz de identificar a casi todos los clientes que efectivamente aceptan, cometiendo muy pocos falsos negativos. Este resultado indica que el modelo está configurado para priorizar la detección de casos positivos por encima de la precisión.

El F1-score de aproximadamente 45%, que combina precisión y recall en una sola métrica, refleja el desequilibrio entre ambas: a pesar de un recall muy alto, la baja precisión limita la armonía del desempeño global. Finalmente, el AUC demuestra que el modelo tiene una buena capacidad discriminativa al asignar probabilidades diferentes a las clases, independientemente del umbral de decisión.

El modelo logra detectar la gran mayoría de los clientes que sí aceptan la oferta, pero al mismo tiempo produce un número considerable de predicciones positivas incorrectas.

## 5. Conclusión

Este comportamiento puede explicarse en parte por la correlación entre variables del dataset, donde algunas presentan asociaciones débiles o redundantes respecto al objetivo. Dichas correlaciones limitan la capacidad del modelo para establecer fronteras de decisión más nítidas, lo que se traduce en predicciones positivas poco precisas. En consecuencia, aunque el modelo logra capturar patrones generales de los clientes que contratan depósitos, los valores de precisión y F1 se mantienen bajos debido a la información poco diferenciadora de ciertas características.

Por lo que los resultados sugieren que para mejorar el desempeño global será necesario refinar la selección de variables, explorar técnicas de balanceo de clases y ajustar el umbral de decisión, de manera que el modelo logre un equilibrio más favorable entre precisión y recall.

El preprocesamiento aplicado resultó fundamental para estabilizar los modelos, el análisis de sesgo y varianza mostró que la regresión logística sufrió de underfitting por su simplicidad del código de la regresión.

## 6. Árboles de decisión

Por ello uno de los mejores modelos para cuando hay poca correlación son los árboles de decisión, que usaremos en esta ocasión para poder predecir las variables y ver cómo sería el mejor modelo, pero para ello primero vamos a explicarlo.

### 6.1 Definición

Los árboles de decisión son modelos de aprendizaje supervisado utilizados tanto para clasificación como para regresión. Su funcionamiento se basa en la división recursiva del espacio de las variables predictoras en regiones cada vez más homogéneas, siguiendo una estructura jerárquica en forma de árbol. En la raíz del árbol se encuentra el conjunto completo de datos. El algoritmo selecciona la variable y el punto de corte que mejor separan las clases (o valores en regresión) según un criterio de impureza.

En cada paso, el árbol busca la división que maximiza la ganancia de información. Este proceso se repite de forma recursiva hasta alcanzar un criterio de parada ya sea profundidad máxima, número mínimo de muestras por hoja o ausencia de ganancia significativa. El resultado es una estructura en forma de árbol donde

- Cada nodo interno representa una condición de decisión sobre una variable.
- Cada rama indica el camino que sigue la instancia según cumpla o no la condición.
- Cada hoja contiene una predicción final: la clase más frecuente o el promedio de valores.

### 6.2 Ventajas

- Interpretabilidad: el modelo puede representarse gráficamente, mostrando claramente cómo se toman las decisiones.
- Flexibilidad: maneja tanto variables numéricas como categóricas, sin necesidad de normalización.
- Captura interacciones: detecta relaciones no lineales entre variables de manera natural.

### 6.3 Desventajas

- Sobreajuste u overfitting: un árbol profundo puede memorizar los datos de entrenamiento y generalizar mal.
- Inestabilidad: pequeños cambios en los datos pueden producir árboles muy distintos.

- Menor precisión individual: suelen ser menos potentes que modelos más complejos.

#### 6.4 Uso en Dataset

En comparación con la regresión logística, los árboles de decisión no dependen de correlaciones lineales entre variables, sino que generan divisiones jerárquicas que permiten detectar reglas más específicas. Sin embargo, al crecer demasiado, pueden alcanzar un accuracy de entrenamiento cercano al 100%, pero con menor desempeño en validación y test, lo que refleja el problema de sobreajuste.

### 7. Preparación de Dataset

Con lo que ya se tenía, la preparación del dataset para el proceso anterior se dejó como estaba, solamente se llevó a cabo la división en conjuntos de entrenamiento y prueba utilizando la función `train_test_split`. Se asignó un 80% de los datos para entrenamiento y un 20% para prueba, lo que permite evaluar el desempeño del modelo en datos no vistos. Se incluyó el parámetro `stratify=y`, con el fin de preservar la proporción original de clases en ambos subconjuntos, evitando sesgos en el balance de la variable objetivo. Además, se fijó `random_state=42` para asegurar la reproducibilidad del proceso.

Después, se verificaron las dimensiones de los conjuntos resultantes, confirmando que los datos quedaron correctamente divididos en entrenamiento y prueba. Este procedimiento aseguró que la información estuviera en un formato limpio, balanceado y adecuado para iniciar la fase de modelado.

### 8. Parámetros del árbol de decisión

Después de la preparación de los datos y su división en conjuntos de entrenamiento y prueba, se procedió a la construcción de un modelo de clasificación basado en Random Forest.

El modelo se definió a través de la clase `RandomForestClassifier` de la librería **scikit-learn**, estableciendo una configuración específica de hiperparámetros:

- `n_estimators=300`: se entrenaron 300 árboles de decisión en el bosque, lo que incrementa la estabilidad y reduce la varianza del modelo.
- `max_depth=None`: se permitió que los árboles crecieran hasta su máxima profundidad posible,

controlando el sobreajuste mediante otras restricciones.

- `min_samples_leaf=5`: cada hoja debe contener al menos 5 muestras, evitando que el modelo genere nodos demasiado pequeños y específicos.
- `min_samples_split=8`: se exige un mínimo de 8 muestras para dividir un nodo, reduciendo el riesgo de sobreajuste.
- `max_features=0.5`: cada árbol selecciona aleatoriamente el 50% de las variables disponibles al realizar una división, fomentando la diversidad entre árboles.
- `max_samples=0.7`: cada árbol se entrena con el 70% de las muestras del conjunto de entrenamiento (submuestreo con reemplazo), reforzando la variabilidad del ensamble.
- `bootstrap=True`: se habilitó el muestreo con reemplazo, técnica característica de Random Forest que genera diferentes subconjuntos de datos para cada árbol.

### 9. Evaluación del Modelo

Tras el entrenamiento del clasificador Random Forest, se procedió a su evaluación sobre el conjunto de prueba. Para ello, se generaron métricas de desempeño que permiten interpretar su capacidad predictiva. Los resultados fueron los siguientes;

Accuracy: 0.8856349114480843				
Reporte de clasificación:				
	precision	recall	f1-score	support
0	0.89	0.99	0.94	7635
1	0.54	0.11	0.18	1004
accuracy			0.89	8639
macro avg	0.72	0.55	0.56	8639
weighted avg	0.85	0.89	0.85	8639

- Accuracy global (88.6%): el modelo logra predecir correctamente la gran mayoría de las instancias en el conjunto de prueba. Este valor refleja un buen desempeño general, aunque debe interpretarse con cautela debido al desbalance de clases existente.
- Clase 0 (clientes que no contratan): presenta un rendimiento sobresaliente, con precisión de 0.89,



recall de 0.99 y F1-score de 0.94. Esto significa que el modelo identifica casi todos los casos negativos y con muy pocos falsos positivos.

- Clase 1 (clientes que sí contratan): el desempeño es significativamente menor, con precisión de 0.54, recall de 0.11 y F1-score de 0.18. Esto indica que, aunque poco más de la mitad de las predicciones positivas resultan correctas, el modelo apenas logra recuperar una fracción de los verdaderos positivos (solo un 11%), lo que refleja una marcada dificultad para detectar la clase minoritaria.
- Macro promedio (macro avg): promediando ambas clases sin ponderar por tamaño, se obtiene un  $\text{precision}=0.72$ ,  $\text{recall}=0.55$  y  $\text{F1}=0.56$ , lo que evidencia el desequilibrio entre clases.
- Promedio ponderado (weighted avg): al ponderar por el número de muestras, los valores son más altos ( $\text{precision}=0.85$ ,  $\text{recall}=0.89$ ,  $\text{F1}=0.85$ ) debido al dominio de la clase negativa.

- Clase 1

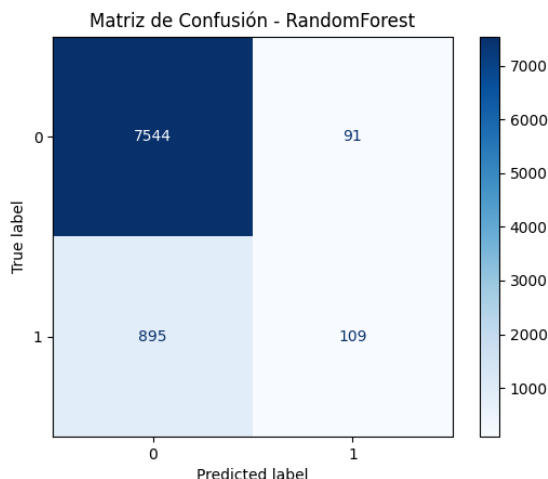
- Falsos Negativos: 895 clientes que sí contrataron, pero el modelo los clasificó erróneamente como negativos.
- Verdaderos Positivos: 109 clientes correctamente identificados como contratantes.

Como vemos El modelo es efectivo para identificar a los clientes que no contratarán siendo 7541 aciertos contra solo 94x errores.

Sin embargo, tiene dificultades para reconocer a los clientes que sí contratan, ya que de 1004 casos positivos, apenas 109 fueron detectados correctamente, mientras que 895 quedaron mal clasificados como negativos.

Este comportamiento está en línea con el reporte de clasificación anterior, donde el recall de la clase positiva fue muy bajo.

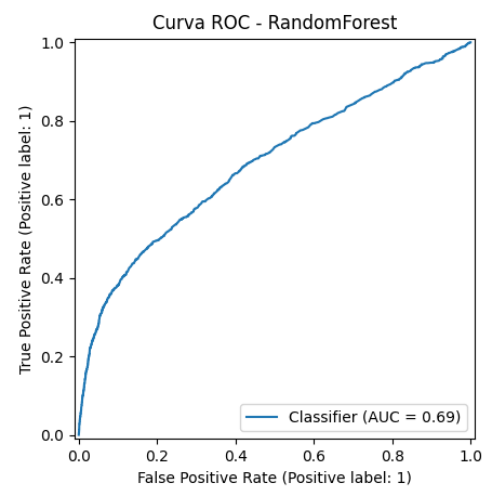
## 8.1 Matriz de Confusión



La matriz de confusión resume el desempeño del modelo mostrando cuántas predicciones fueron correctas o incorrectas en cada clase. En este caso:

- Clase 0:
  - Verdaderos Negativos: 7544 clientes correctamente identificados como no contratantes.
  - Falsos Positivos: 91 clientes que realmente no contrataron, pero el modelo los predijo como si sí lo hicieran.

## 8.2 Curva ROC



Un  $\text{AUC}=0.69$  es un valor moderado: mejor que un clasificador aleatorio, pero lejos de un modelo óptimo.

La curva muestra que, para obtener un recall alto que detectar casi todos los positivos, el modelo incurre en un aumento significativo de falsos positivos.

Por ello se va a modificar el modelo para que pueda mejor



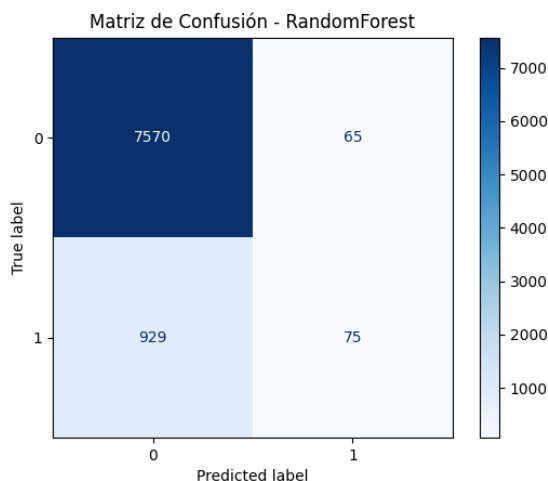
## 10. Mejora de Modelo

Con el objetivo de mejorar el desempeño del modelo y reducir el sobreajuste observado previamente, se construyó un segundo clasificador basado en Random Forest con una configuración más restrictiva de los hiperparámetros.

El modelo rf\_2 se definió con los siguientes valores:

- `n_estimators=500`: se incrementó el número de árboles a 500, lo que proporciona mayor estabilidad al ensamble y reduce la varianza en las predicciones.
- `max_depth=8`: se limitó la profundidad máxima de los árboles a 8 niveles, controlando la complejidad del modelo e impidiendo que memorice patrones demasiado específicos de los datos de entrenamiento.
- `min_samples_leaf=4`: cada hoja debe contener al menos 4 instancias, lo que favorece divisiones más generales y evita nodos terminales basados en muy pocos datos.
- `min_samples_split=10`: se estableció un mínimo de 10 instancias para dividir un nodo, reduciendo la tendencia a generar particiones excesivamente pequeñas.

### 10.1 Matriz de Confusión



La matriz de confusión muestra cómo se distribuyen los aciertos y errores del modelo en el conjunto de prueba:

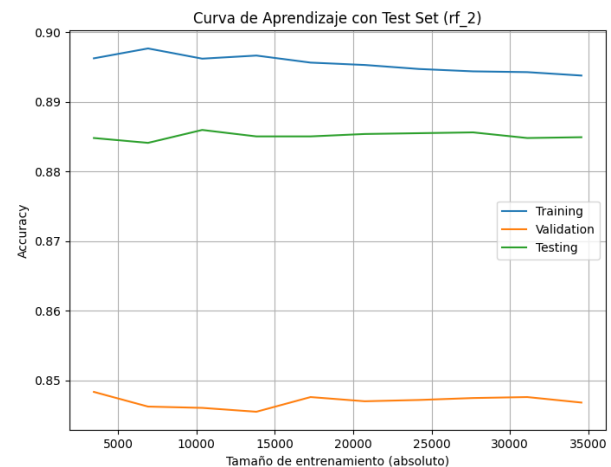
- Clase 0 (clientes que NO contratan):

- Verdaderos Negativos 7570: clientes correctamente identificados como no contratantes.
- Falsos Positivos 65: clientes que realmente no contrataron, pero fueron clasificados como si sí lo hicieran.

- Clase 1 (clientes que SÍ contratan):

- Falsos Negativos 929: clientes que sí contrataron, pero fueron clasificados erróneamente como negativos.
- Verdaderos Positivos 75: clientes correctamente identificados como contratantes.

### 10.2 Curva de Aprendizaje



La gráfica presenta la evolución de la accuracy en los conjuntos de entrenamiento, validación y prueba conforme aumenta el tamaño de los datos de entrenamiento.

- Conjunto de entrenamiento:  
Se mantiene alto y estable lo que indica que el modelo logra aprender correctamente los patrones de la clase mayoritaria sin sobreajustarse demasiado. El leve descenso con más datos es esperado, ya que se vuelve más difícil memorizar todos los ejemplos.
- Conjunto de validación:  
Se sitúa alrededor de 0.84–0.85 y muestra estabilidad a lo largo de todos los tamaños de entrenamiento. Esto indica que el modelo tiene un

desempeño constante y que aumentar datos no genera cambios significativos en la validación.

- Conjunto de prueba:  
Se ubica en torno a 0.88–0.89, ligeramente por debajo del entrenamiento pero por encima de la validación. Esto sugiere que el modelo logra generalizar bien hacia datos no vistos, aunque conserva cierta dificultad en capturar completamente la clase minoritaria.

## 11. Análisis de Sesgo y Varianza

- Regresión Logística: presentó sesgo medio-alto y varianza baja. Esto refleja un *underfitting*, ya que el modelo es demasiado simple para los patrones del dataset.
- Random Forest (modelo base): mostró sesgo bajo, pero varianza alta, con alta precisión en entrenamiento, pero menor en validación, lo que indica un *overfitting*.
- Random Forest ajustado: alcanzó un equilibrio con sesgo medio y varianza baja-media, manteniendo métricas estables en validación y prueba. Esto corresponde a un modelo bien ajustado, que logra generalizar sin memorizar los datos de entrenamiento.

Este análisis se refleja en los resultados: la regresión logística, bajo el umbral estándar de 0.5, mostró un recall de apenas 0.001, lo que evidencia un sesgo alto hacia la clase negativa. Tras ajustar el umbral a 0.3, el recall aumentó hasta 0.861, reduciendo el sesgo, pero a costa de menor precisión. En contraste, el Random Forest base tuvo gran desempeño en entrenamiento, pero decayó en validación, confirmando varianza alta. El modelo ajustado, con validación estable en 0.85 y prueba en 0.88, logró un equilibrio adecuado entre sesgo y varianza, representando un modelo bien ajustado

## 12. Conclusión

El desarrollo de este proyecto permitió comprender de manera integral el ciclo de construcción y evaluación de modelos de clasificación, aplicando buenas prácticas de preprocesamiento, validación y regularización. En la etapa de preparación de datos, se implementaron al menos dos técnicas clave: la imputación por moda en variables categóricas con valores desconocidos (*job* y *education*) y el escalado robusto de variables numéricas con alta presencia de outliers, como *balance* y *duration*. Estas

decisiones resultaron fundamentales para reducir sesgos, estabilizar el entrenamiento y garantizar que las variables tuvieran un rango comparable.

La evaluación del desempeño se realizó con un diseño experimental sólido que incluyó separación en entrenamiento, validación y prueba, además de validación cruzada estratificada en 5 pliegues. Este enfoque permitió estimar de forma confiable la capacidad de generalización de los modelos y reducir el riesgo de sobreajuste en la fase de ajuste de hiperparámetros. El análisis de las curvas de aprendizaje demostró la relevancia de observar tanto el error de entrenamiento como el de validación para diagnosticar el comportamiento de sesgo y varianza.

Los resultados evidenciaron comportamientos contrastantes. La regresión logística mostró un **sesgo medio-alto y varianza baja**, reflejo de un **underfitting**, al ser un modelo lineal que no captura adecuadamente la complejidad del problema. El Random Forest en su configuración base presentó **sesgo bajo y varianza alta**, lo que derivó en **overfitting**: alto desempeño en entrenamiento pero menor capacidad de generalización. Finalmente, el Random Forest ajustado con hiperparámetros de regularización alcanzó un **equilibrio adecuado (fit)**, con sesgo medio y varianza baja-media, logrando un rendimiento estable y consistente en validación y prueba. Estas observaciones permitieron comprender cómo diagnosticar el grado de bias y varianza y cómo relacionarlos con el nivel de ajuste del modelo.

El uso de técnicas de regularización fue esencial para mejorar el desempeño, como en Random Forest, la limitación de profundidad, el tamaño mínimo de hojas, el fraccionamiento de muestras y el submuestreo de características y registros funcionaron como mecanismos efectivos de control de la varianza, esto permitió mejorar la capacidad de generalización y consolidaron el aprendizaje de cómo aplicar regularización de manera estratégica en distintos algoritmos.