PREDICCIÓN DEL RIESGO DE DEPRESIÓN EN ESTUDIANTES UNIVERSITARIOS MEDIANTE MÉTODOS DE APRENDIZAJE AUTOMÁTICO

Kassandra Mendoza Sarmiento
Código 9024393
Ángel David Sánchez Serna
Código 9014957
Juan Sebastián Cuchumbé Mera
Código 9014663

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Director
PhD. José Hernando Ávila-Toscano

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, JUNIO DE 2026

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

Durante mucho tiempo, las enfermedades mentales no recibieron la atención que merecían, sin embargo, en años recientes la Organización Mundial de la Salud (OMS) las ha reconocido como un problema de salud pública, estimando que afectan entre el 3% y el 5% de la población mundial [1]. Una de las afectaciones más comunes es la depresión, cuyos síntomas incluyen aburrimiento, tristeza profunda, desmotivación, agotamiento físico y mental, y, en el peor de los casos, puede llevar al suicidio. Hallazgos recientes indican que los estudiantes universitarios son una de las poblaciones donde esta afección es más prevalente, lo cual puede asociarse con la exposición continua al estrés debido a sus responsabilidades académicas, relaciones personales, familiares, laborales y económicos [2].

Sumado a lo anterior, la mayoría de las instituciones de educación superior no ofrece programas o herramientas orientadas a la prevención de este trastorno [3], ya sea por limitaciones de recursos o porque, en algunos casos, los mismos estudiantes perciben que estas ayudas están dirigidas únicamente a personas con trastornos mentales más graves.

Con base en lo expuesto, en este proyecto aplicado final de maestría, se desarrolló un modelo de análisis que permitió predecir el riesgo de depresión en estudiantes universitarios, brindando a las instituciones herramientas que ayudan a prevenir de manera oportuna los posibles casos de depresión. Para ello, se construyeron dos modelos predictivos: el primero se basó en regresión logística, con el fin de analizar la relación entre variables desde un enfoque lineal. El segundo empleó el algoritmo Random Forest Classifier, el cual permitió capturar interacciones no lineales y evaluar la importancia relativa de cada predictor. Este ejercicio se realizó a partir de datos públicos contenidos en el Student Depression Dataset, disponible en la plataforma Kaggle.

El objetivo fue estimar el riesgo de depresión en estudiantes de educación superior considerando variables sociodemográficas, emocionales y académicas, ya que estos factores influyen significativamente en la aparición de trastornos mentales dentro de esta población [4]. Como resultados, a partir de este estudio se estimaron y validaron ambos modelos, lo que permitió identificar los predictores más relevantes; además, se obtuvo un análisis comparativo del rendimiento de los modelos a partir del cálculo de métricas como la precisión, la sensibilidad, la especificidad y la capacidad de los modelos para discriminar entre clases, distinguiendo en clase positiva y negativa, esto mediante el cálculo del área bajo la curva (AUC).

1 CONTEXTUALIZACIÓN DEL PROYECTO

1.1 Definición del problema

1.1.1 Planteamiento del problema

La depresión es un trastorno mental de alta prevalencia que implica pérdida de placer o interés en actividades, lo cual puede afectar todos los aspectos de la vida, incluidas las relaciones con la familia, amigos y comunidad en general. Según la Organización Mundial de la Salud (OMS), a nivel mundial se estima que el 3.8% de la población sufre depresión [5]. Otros estudios [6] señalan que entre 1990 y 2021, la carga mundial del trastorno depresivo mayor aumentó. La incidencia incrementó un 56.1% y la prevalencia un 56.36%, particularmente entre hombres y adultos jóvenes de 20 a 24 años. Después de 2020, las métricas (incidencia, prevalencia y Años de Vida Ajustados por Discapacidad - AVAD) tuvieron un aumento drástico, posiblemente debido a las medidas de confinamiento prolongados, aislamiento social e interrupciones en los servicios de salud mental implementadas durante la pandemia de COVID-19 [6].

La depresión puede afectar a más del 20% de los jóvenes en algún momento de su vida, y contribuye con un 8.2% de la carga global de enfermedad en niños y adolescentes de 10 a 24 años [7]. En jóvenes, puede manifestarse con síntomas como irritabilidad, agresión, problemas escolares, fatiga, apatía, cambios de apetito (por aumento o por defecto), falta de motivación y ansiedad.

En los últimos años, el interés por la salud mental de la población joven ha penetrado en el escenario educativo, al punto de transformarse en una preocupación para la comunidad académica representada por los estudiantes universitarios. Los reportes internacionales han mostrado la elevada prevalencia de síntomas depresivos que afectan tanto su bienestar emocional como también su rendimiento académico [8]. Adicionalmente, existen estudios que relacionan la depresión con factores como la ansiedad, el estrés y los problemas en el rendimiento académico [9].

Según la OMS, pese a que existen tratamientos conocidos y eficaces para los trastornos mentales, en países de bajos y medianos ingresos más del 75% de las personas no reciben tratamientos debido a la falta de inversión en salud mental, falta de profesionales capacitados y el estigma social asociado con los trastornos mentales [5]. Esta realidad también se refleja en las instituciones universitarias, que pese a contar con áreas de bienestar y programas de acompañamiento emocional, suelen estar más diseñados para reaccionar cuando ocurre un evento adverso producto del estado emocional del estudiante, es decir, no se suele tener herramientas que permitan identificar anticipadamente estudiantes en riesgo.

Como lo señala el Departamento de Educación de los Estados Unidos, uno de los desafíos clave frente a esta problemática consiste en superar la implementación ineficaz de prácticas de prevención basadas en evidencia, lo cual refleja que muchas estrategias actuales están más enfocadas en responder después de que los problemas aparecen, en lugar de prevenir [10]. Lo anterior, además de afectar la capacidad de intervención temprana para los estudiantes, también afecta la permanencia estudiantil. En este contexto, es fundamental que las Instituciones de Educación Superior (IES) cuenten con un sistema integral de apoyo a la salud mental que incluya tanto estrategias preventivas como

intervenciones reactivas y referencia a servicios especializados para casos complejos [11].

Se requieren herramientas que permitan la identificación temprana de estudiantes en riesgo de depresión, mediante las cuales se beneficien los estudiantes y se impacte positivamente en la permanencia académica reduciendo la deserción universitaria. La aplicación del aprendizaje automático sobre los programas de bienestar estudiantil brinda información basada en datos que permite orientar las estrategias de prevención e intervención [12]. En este sentido, la ciencia de datos genera oportunidades significativas para realizar modelos predictivos que identifiquen anticipadamente factores de riesgo. Técnicas como la regresión logística y Random Forest Classifier (RFC) permiten analizar conjuntamente variables emocionales, académicas y sociodemográficas que proporcionan información valiosa para la toma de decisiones en las áreas de bienestar estudiantil de las instituciones universitarias.

El uso de técnicas basadas en aprendizaje automático o *Machine Learning* (ML), permite modelar estadísticamente problemas complejos como el riesgo de depresión en estudiantes universitarios superando los enfoques estadísticos tradicionales. El modelo RFC permite trabajar con datos que no cumplen supuestos clásicos como la normalidad, además de detectar relaciones no lineales que las regresiones tradicionales no capturan. Este método también permite evaluar la validez de los modelos mediante técnicas de validación cruzada y métricas de desempeño como el AUC-ROC la precisión y el índice de *Youden*, este último útil para determinar umbrales óptimos de clasificación en contextos de salud mental [13]. Lo anterior es de gran utilidad cuando se trabaja con datos heterogéneos como las variables psicológicas y sociodemográficas. La capacidad predictiva de estos modelos abre nuevas posibilidades para la prevención temprana de problemáticas como la depresión y la deserción universitaria.

1.1.2 Formulación del problema

Este estudio busca dar respuesta a la siguiente pregunta de investigación aplicada:

¿Cómo se puede estimar el riesgo de depresión en estudiantes universitarios mediante el uso de métodos de aprendizaje automático basados en variables sociodemográficas, emocionales y académicas?

1.1.3 Sistematización

La respuesta a la pregunta anterior promueve la inclusión y permanencia educativa mediante el uso ético y aplicado de modelos estadísticos. Esto implica responder las siguientes preguntas:

¿Qué capacidad posee la regresión logística para estimar el riesgo de depresión en universitarios a partir de relaciones lineales entre variables sociodemográficas, emocionales y académicas?

¿Cómo se desempeña el modelo Random Forest Classifier en la predicción del riesgo de depresión, identificando patrones no lineales y la importancia relativa de los predictores?

¿Cuál de los modelos presenta mejor rendimiento en las estimaciones predictivas de depresión de acuerdo con métricas de precisión, sensibilidad y especificidad?

1.2 Objetivos

1.2.1. Objetivo General

Estimar el riesgo de depresión en estudiantes universitarios a partir de variables sociodemográficas, emocionales y académicas, utilizando modelos de aprendizaje automático como regresión logística y Random Forest Classifier.

1.2.2. Objetivos Específicos

- Analizar la capacidad de la regresión logística para estimar el riesgo de depresión en universitarios y describir relaciones lineales entre variables sociodemográficas, emocionales y académicas.

- Evaluar el desempeño del modelo Random Forest Classifier al predecir el riesgo de depresión, identificando patrones no lineales y la importancia relativa de los predictores.

- Comparar el rendimiento predictivo de ambos modelos de acuerdo con métricas de precisión, sensibilidad y especificidad, para determinar cuál proporciona estimaciones más robustas del riesgo de depresión.

1.3 Marco de Referencia

1.3.1. Marco Teórico

1.3.1.1 Salud mental en estudiantes universitarios y factores de riesgo

La salud mental es definida por la Organización Mundial de la Salud (OMS) como un estado de bienestar en el cual el individuo puede desarrollar sus capacidades, afrontar las tensiones normales de la vida, trabajar de forma productiva y contribuir a su comunidad [14]. Este concepto implica la ausencia de trastornos mentales, así como el reconocimiento de que existen factores que favorecen el equilibrio emocional, social y psicológico. En el contexto universitario, esta definición cobra relevancia debido a que los estudiantes enfrentan desafíos que pueden afectar su bienestar, como altos niveles de exigencia académica, la gestión del tiempo y los aspectos sociales que pueden incidir en la salud mental.

A nivel global, se ha documentado una alta prevalencia de síntomas relacionados con trastornos mentales en estudiantes universitarios, particularmente depresión y ansiedad. Un metaanálisis que incluyó 64 estudios con más de 100.000 universitarios de diversas regiones encontró que la prevalencia global de síntomas depresivos fue del 33.6% y la de ansiedad del 39% [15]. Estos porcentajes son especialmente altos en estudiantes de medicina y en países de ingresos bajos y medianos.

En el contexto colombiano, diversos estudios han evidenciado una realidad similar. Una investigación transversal realizada en Bucaramanga con 1957 estudiantes durante el confinamiento por COVID-19,

encontró que 2.5% de los estudiantes universitarios eran casos clínicos definitivos de ansiedad y 8.2% de depresión. El análisis evidenció una asociación entre los síntomas de salud mental y el bajo rendimiento académico, con mayor prevalencia entre las mujeres. En particular, las estudiantes clasificadas como "casos dudosos" de depresión presentaban el doble de probabilidad de bajo desempeño académico (*PR* = 2.0; *IC95*% = 1.10, 5.18; *p* = .03), mientras que los hombres con bajo rendimiento presentaban una prevalencia de casos definitivos de depresión del 12.2% (*p* < .05) [16].

Otro estudio observacional que examinó la relación entre el capital social, los estilos de vida y los síntomas depresivos en 609 estudiantes universitarios colombianos, encontró que 22.4% de los participantes presentaban síntomas depresivos clínicamente relevantes [17] según la escala CES-D (*Center for Epidemiologic Studies Depression Scale*), un instrumento validado para medir la sintomatología depresiva en estudios poblacionales [18].

El análisis multivariado mostró que los estudiantes con niveles bajos de capital social estructural presentaban una razón de prevalencia (PR) de 2.14 (*IC95%* = 1.56, 2.93; *p* < .001) para síntomas depresivos, en comparación con quienes tenían niveles altos de capital social. Asimismo, se evidenció que prácticas como el sedentarismo y la escasa interacción social estaban asociadas con un mayor deterioro del bienestar psicológico. Durante la pandemia y el aislamiento por COVID-19, el porcentaje de estudiantes que reportaron pasar más de 10 horas sentados al día aumentó en un 31.49%. Paralelamente, los niveles de capital social cognitivo y conductual disminuyeron en un 12.03% y 24.54%, respectivamente. Estas condiciones se correlacionaron con un incremento en la prevalencia de síntomas depresivos clínicamente relevantes, siendo particularmente importante la asociación con bajos niveles de capital social conductual (*OR* = 1.88; *IC95%* = 1.16, 3.04) [17].

Además, según un estudio realizado a 6224 universitarios de una institución privada colombiana, se identificaron diversos factores asociados con mayores niveles de ansiedad y depresión, entre los que destacan la sobrecarga académica, el escaso apoyo social, las dificultades económicas, las alteraciones en el patrón de sueño y el consumo problemático de sustancias psicoactivas. Estos factores muestran una correlación con los niveles de ansiedad y depresión, evidenciando que 48.3% de la varianza en los puntajes de ansiedad y 42. % en los de depresión, pueden ser explicados por estas variables psicosociales [19].

1.3.1.2 Depresión y entorno universitario

La depresión se caracteriza por la presencia persistente de tristeza, pérdida de interés o placer, disminución de la energía, dificultades de concentración, sentimientos de culpa o baja autoestima, y alteraciones (por exceso o por defecto) del sueño o del apetito. Según la Organización Mundial de la Salud (OMS), más de 280 millones de personas en el mundo la padecen, lo que la convierte en una de las principales causas de discapacidad a nivel global. Además, afecta significativamente el funcionamiento social, laboral y académico de quienes la sufren [20].

En el entorno universitario, los síntomas depresivos pueden expresarse a través de conductas como el aislamiento social, la falta de motivación académica, la procrastinación, el ausentismo y la pérdida de interés en actividades previamente gratificantes. Estos signos suelen ser minimizados o atribuidos erróneamente a estrés académico, lo que hace difícil una identificación oportuna [19].

En el contexto de este proyecto, la depresión se entiende como un estado afectivo complejo, cuya aparición puede estar vinculada a múltiples factores psicosociales, académicos y personales. El objetivo del modelo predictivo no es realizar diagnósticos clínicos, sino identificar patrones de riesgo que permitan alertar a las instituciones educativas sobre posibles casos que requieran atención psicosocial especializada, contribuyendo así a la prevención y el acompañamiento oportuno.

1.3.1.3 Predicción de trastornos mentales

El uso de modelos predictivos en el campo de la salud mental ha ganado relevancia en los últimos años, incluyendo entornos educativos donde se ha hecho más importante identificar tempranamente a estudiantes en riesgo de padecer algún trastorno. Estos modelos, fundamentados en técnicas de aprendizaje automático, permiten analizar variables sociodemográficas, académicas y comportamentales para predecir la probabilidad de desarrollar síntomas depresivos o ansiosos, entre otros. Una revisión sistemática reciente destaca el uso frecuente de algoritmos como la regresión logística, árboles de decisión, Random Forest, máquinas de vectores de soporte (*Support Vector Machines*, SVM) y redes neuronales, los cuales han mostrado desempeños aceptables en la clasificación de trastornos mentales, con precisiones que superan el 80% en algunos estudios [21].

El aprendizaje automático (*Machine Learning*) es una subdisciplina de la inteligencia artificial que se basa en el desarrollo de algoritmos capaces de identificar patrones en grandes volúmenes de datos y realizar predicciones sin ser explícitamente programados para cada tarea. Según Mitchell y Jordan (2015), un programa de computador se considera que aprende de la experiencia E, con respecto a una clase de tareas T y una medida de desempeño P, si su desempeño en las tareas de T, medido por P, mejora con la experiencia [22].

En el área de la salud mental, esta tecnología ha sido aplicada para anticipar la aparición de trastornos como la depresión o la ansiedad, especialmente en poblaciones universitarias, donde la prevalencia de sintomatología psicológica es considerable. Entre los enfoques más comunes se encuentran los modelos de aprendizaje supervisado, en los que el algoritmo se entrena utilizando datos etiquetados. Este tipo de aprendizaje es adecuado para abordar problemas de clasificación binaria, como estimar la probabilidad de que un estudiante presente riesgo de desarrollar síntomas depresivos.

Dos algoritmos frecuentemente empleados en este contexto son:

Regresión logística: modelo estadístico que estima la probabilidad de ocurrencia de un evento binario (por ejemplo, presencia o ausencia de síntomas depresivos) a partir de una combinación lineal de variables independientes. Se destaca por su interpretabilidad, lo cual permite comprender el efecto de cada variable sobre la probabilidad estimada [23].

La regresión logística se puede ampliar por medio de las técnicas de regularización, cuyo objetivo es hacer frente al sobreajuste (ajuste del modelo extremadamente bien en la muestra, pero mal en la población) y mejorar la capacidad de generalización del modelo, favoreciendo así la capacidad de predecir con mayor precisión. Estas técnicas están compuestas por: Lasso (L1), Ridge (L2) y Elastic net, las cuales realizaran una penalización sobre los coeficientes del modelo, favoreciendo casos donde se presentan muchas variables predictoras [31].

Por otro lado, el modelo logístico puede abordarse desde un enfoque predictivo o explicativo, dependiendo del objetivo de análisis, lo que permite observar los resultados de las variables predictoras como la probabilidad que ocurrencia de un evento.

Random Forest Classifier: método de ensamble que construye múltiples árboles de decisión sobre subconjuntos aleatorios del conjunto de datos y toma la decisión final mediante votación. Este enfoque presenta buena capacidad de generalización, es robusto frente al sobreajuste y puede manejar tanto variables numéricas como categóricas [24]. Este es un método no paramétrico que no asume supuestos de linealidad o independencia entre sus variables predictoras, lo que permite manejar adecuadamente relaciones complejas y escenarios con multicolinealidad [30].

Además, la evaluación del desempeño de los modelos clasificadores se realiza mediante métricas específicas, entre las que se destacan:

- Accuracy (exactitud): proporción de predicciones correctas sobre el total de casos.
- Recall (sensibilidad): proporción de verdaderos positivos sobre el total de casos positivos reales.
- F1-score: media armónica entre precisión y sensibilidad.
- AUC-ROC: área bajo la curva que relaciona la tasa de verdaderos positivos con la de falsos positivos.

Estas métricas permiten comparar objetivamente el rendimiento de diferentes modelos y seleccionar el más adecuado de acuerdo con los objetivos del proyecto y la sensibilidad del contexto en el que se aplicará [25].

1.3.1.4 Formulación de las métricas e indicadores empleados

Para precisar las definiciones empleadas en la evaluación de los modelos, se incluyen a continuación las fórmulas de las principales métricas e indicadores. En ellas, VP corresponde a verdaderos positivos, VN a verdaderos negativos, FP a falsos positivos y FN a falsos negativos.

- $Sensibilidad\ (recall) = \frac{VP}{(VP+FN)}$

- $Especificidad = \frac{VN}{(VN+FP)}$

- $Precisión = \frac{VP}{(VP+FP)}$
- $F1 - score = 2 \times \frac{(Precisión \times Sensibilidad)}{(Precisión + Sensibilidad)}$
- $Brier\ Score = \frac{1}{n} \times \sum (P_i - Y_i)^2$

$$Donde\ P_i\ es\ la\ probabilidad\ predicha\ y\ Y_i\ el\ valor\ observado\ (0\ o\ 1)$$

- $\text{Índice de Youden } (J) = Sensibilidad + Especificidad - 1$
- $\text{Función logística } P(Y = 1|x) = \frac{1}{(1+ e^{-(\beta_0+\beta_1 X_1+..+ \beta_k X_k)})}$

- $Odds\ Ratio(OR) = e^{\beta}$, donde β es el coeficiente estimado de la variable correspondiente.

- $VIF_j = \frac{1}{(1- R_j^2)}$, Donde R_j^2 es el coeficiente de determinación de la regresión de la variable j sobre el resto de predictores

1.3.2. Antecedentes

En trabajos o estudios previos [1], se ha demostrado que la depresión afecta en su mayoría a la población universitaria, principalmente producto de que deben cumplir con varias responsabilidades como: diversos cursos, plazos de entrega, exámenes, responsabilidades con sus familias, pareja y en algunos casos, compromisos económicos.

Algunas investigaciones recientes han empleado variables o datos que permitan prevenir este trastorno, ya que la mayoría de los estudios se centran únicamente en abordarlo, sin profundizar suficientemente en las causas que lo originan. Un trabajo anterior [26], analizó los distintos tipos de trastornos mentales y características de la población estudio, por medio de modelos como Machine learning, Random Forest, Gradiant booster Classifier, entre otros, mediante los cuales se buscó predecir la patología mental, con el fin de dar el mejor tratamiento para evitar así el suicidio por estos trastornos mentales.

Por otro lado, el estudio realizado por Amú, González, Ortiz y Sandoval [29], empleó un modelo predictivo basado en árboles de decisión en una universidad pública colombiana, con una muestra de 1059 estudiantes. Este trabajó tuvo como objetivo identificar el riesgo de que los estudiantes presentaran o desarrollaran estrés. Los resultados mostraron una sensibilidad de .83, una especificidad de .72 y una precisión de .72. Con base en estos hallazgos, los autores concluyeron que el modelo de Random Forest puede predecir o anticipar el comportamiento de estrés en estudiantes universitarios.

Finalmente, la predicción de cuadros depresivos y otros problemas de salud mental mediante modelos de *machine learning* reviste gran importancia, ya que la literatura ha mostrado la utilidad de estas herramientas para la detección oportuna de esta problemática [28]. En este sentido, pueden constituirse en un insumo crucial para que las instituciones de nivel superior adopten medidas preventivas orientadas a reducir las consecuencias asociadas con la depresión como el suicidio, la deserción estudiantil, y otras afectaciones al bienestar de los estudiantes.

2 METODOLOGIA

2.1 Materiales

El proyecto se realizó por medio de equipos de cómputo, utilizando Google colab para la elaboración de los dos modelos (lineal y random forest). También se utilizó una base de datos llamada *Student Depression Dataset,* a la cual se accedió de forma libre y gratuita a través del sitio web Kaggle. El conjunto de datos utilizado contaba con información sobre estudiantes universitarios los cuales, fueron recolectados mediante encuestas para medir factores asociados al riesgo de depresión. El total de observaciones de este dataset era de 27892 con 13 variables, con características sociodemográficas (edad, genero), factores académicos (presión académica, satisfacción con los estudios, horas de trabajo o estudio), hábitos diarios (horas de sueño, hábitos alimenticios), estado emocional (estrés financiero, ideas suicidas), antecedentes familiares y un indicador de depresión. Estas variables brindaron un contexto académico, emocional y personal de los estudiantes que participaron en la encuesta.

El flujo de preparación de datos fue implementado en Python mediante un cuaderno reproducible de Google Colab. La carga del dataset, definición de variable objetivo, limpieza de registros, selección del conjunto principal de predictores y construcción del pipeline de preprocesamiento se presentan en el **Anexo B, Código B1**.

2.1.1 Objetivo específico 1. Modelo de regresión logística.

Para dar alcance al objetivo específico 1 se elaboró un modelo de regresión logística binaria explicativo, con el fin de estimar la probabilidad de que un estudiante presente síntomas depresivos. El usó de este modelo se diferencia del enfoque predictivo en los objetivos posteriores, ya que, se busca maximizar el desempeño del modelo y no en la interpretación de coeficientes. Este proceso inició con los datos depurados (*df_clean*), lo cual permitió definir el modelo principal con las variables predictoras, pero retirando aquellas variables similares a la variable objetivo.

Una vez obtenida esta base de datos estable para el modelo, se construyó una matriz de predictores y el vector objetivo. Haciendo uso de las librerías pandas, se identifican las variables categorías y numéricas para así aplicar el tratamiento correcto para cada variable. Las variables categóricas fueron tratadas con la técnica *one-hot encoding* por medio de la función *get_dummies()*. Con el fin de evitar la colinealidad asociada a ciertas variables, se eliminó la primera categoría de cada variable categórica porque, actuaban como categoría de referencia en la estimación del modelo. Así, se eliminó la primera categoría para evitar la colinealidad.

Posteriormente, se evaluó la multicolinealidad entre predictores mediante el Variance Inflation Factor (VIF) con el fin de inspeccionar la estabilidad de las estimaciones. Luego se ajustó el modelo de regresión logística utilizando la función Logit de la librería *statsmodels*. Con este modelo ya ajustado, se obtuvieron los coeficientes, errores estándar, valores z, valores p, intervalos de confianza del 95% y los odds ratios, lo cual permitió identificar el efecto de cada predictor en la variable objetivo.

Por último, se calcularon los indicadores globales de ajuste log-likelihood, el pseudo R^2 de McFadden, así como las métricas AIC y BIC para evaluar la efectividad del modelo. Todos los resultados generados se

exportaron para su documentación y análisis posterior.

2.1.2 Objetivo específico 2. Modelo Random Forest Classifier.

Para dar cumplimiento al objetivo específico 2 se aplicó la técnica Random Forest Classifier para detectar el riesgo de padecer depresión en la población estudiantil universitaria. Para esto, se utilizó el mismo grupo de variables predictores del modelo anterior, con el fin de garantizar consistencia que luego permitiera realizar la comparación entre modelos en el siguiente objetivo.

Se tomó el 80% de los datos (n = 22.313) tratados para entrenar el modelo y el 20% restante (n = 5578) para realizar la evaluación del modelo, esto con el fin de validar el método de aprendizaje del modelo. Por medio de la función *Columntransformer*, se garantizó un tratamiento según el tipo de variables: las numéricas fueron tratadas con la función *passthrough;* las categóricas se trataron con la función *OneHotEncoder.* Esto permitió eliminar la primera categoría de cada variable categórica para evitar la colinealdiad propia del one-hot encoding, de esta manera, se habilitó el manejo de categorías no observadas por la función handle_unkown=" ignore" esto permitió el procesamiento adecuado de valores nuevos en el conjunto de prueba por parte del modelo.

Posteriormente se empleó la función *StratifiedKFold* para realizar validación cruzada estratificada con aleatoriedad controlada, con el objetivo de preservar la proporción de clases en los pliegues. Posteriormente, se tomó este esquema para realizar la búsqueda de hiperparámetros mediante la función *GridsearchCv*, la cual media la combinación de árboles, criterios de división y proporción de predictores. Por medio del área bajo la curva ROC (AUC) como métrica principal, se eligió el modelo con mejor capacidad discriminativa promedio.

Al seleccionar el mejor grupo de hiperparámetros, el modelo final fue ajustado y se extrajeron las importancias relativas de los predictores, lo cual permitió identificar aquellos que influían en mayor medida en el modelo. Luego, se generaron las probabilidades predichas con el conjunto de datos de prueba, las cuales fueron organizadas en tablas y documentadas para su análisis.

Por último, el modelo fue evaluado con las métricas globales, matrices de confusión y curvas ROC; este conjunto de evaluaciones permitió caracterizar la capacidad predictiva del modelo Random Forest y establecer su utilidad comparativa frente al modelo logístico regularizado.

2.1.3 Objetivo específico 3. Comparación entre modelos.

El desarrollo del objetivo específico 3 consistió en comparar el desempeño predictivo de los modelos Ridge y Random Forest Classifier bajo las mismas condiciones de evaluación. Para garantizar una comparación justa, ambos modelos fueron entrenados y probados utilizando exactamente la misma partición de datos, las mismas variables predictoras y el mismo conjunto de métricas. De esta manera, cualquier diferencia observada en el desempeño puede atribuirse al método de modelado y no a variaciones en los datos o en el proceso de validación.

En el modelo de regresión logística Ridge se realizó un proceso de selección del hiperparámetro de regularización C, con el cual se controla la penalización aplicada a los coeficientes. Para esto, se utilizó una

validación cruzada estratificada de cinco pliegues sobre los datos de entrenamiento, evaluando 13 valores de C distribuidos en escala logarítmica entre 0.001 y 1000. Los valores de AUC promedio y su desviación estándar se registraron por cada valor, para identificar la configuración estable y mejor capacidad discriminativa. Estos resultados fueron ordenados y documentando en la sección de resultados.

Esta comparación se hizo con tres componentes, primero se compararon las métricas globales de ambos modelos, después se comparó los umbrales de clasificación y, por último, las curvas ROC. En las métricas globales, se organizaron y compraron los siguientes valores: AUC de entrenamiento y prueba, precisión promedio y Brier score; la métrica Brier score se incluyó porque evalúa el grado en que las probabilidades predichas reflejan acertadamente la frecuencia real del evento, esto permite estimar el riesgo individual y no solo discriminar entre clases.

Mientras que los umbrales de clasificación consistieron en tomar los valores de: umbral, sensibilidad, especificidad, precisión, F1-score, FN y FP, además de esto se incorporó el índice de *Youden* el cual permitió identificar el punto de la curva de ROC que maximiza la detección correcta de los casos positivos y negativos. El índice de *Youden*, se utilizó para calcular el umbral optimo con base en los datos de entrenamiento, y usado como criterio de referencia al comparar los modelos junto los valores de las curvas ROC.

Una vez organizada la información, los resultados se compararon entre sí, para seleccionar el modelo que mejor desempeño obtuvo.

3 RESULTADOS

3.1 Descripción general de base de datos Student Depression Dataset

La base de datos *Student Depression Dataset* está compuesta por 27901 registros y 18 variables recolectadas a través de encuestas aplicadas a estudiantes y contiene información sociodemográfica, académica, de hábitos de vida y de salud mental. En la Tabla 1 se muestran las variables, junto con su descripción y tipo.

Tabla 1: Variables, su descripción y tipo

Código de la variable	Traducción de la variable	Descripción	Tipo de variable
id	ID	Identificador único del registro	Tipo_ID
gender	Género	Género del estudiante (Male, Female)	Categórica nominal
age	Edad	Edad en años	Numérica continua
city	Ciudad	Ciudad de residencia	Categórica nominal
profession	Profesión	Ocupación actual (principalmente "Student")	Categórica nominal
academic_pressure	Presión académica	Nivel de presión académica (0–5)	Numérica ordinal
work_pressure	Presión laboral	Nivel de presión laboral (0–5)	Numérica ordinal
cgpa	Promedio académico	Promedio académico (0–10)	Numérica continua
study_satisfaction	Satisfacción con el estudio	Satisfacción con los estudios (0–5)	Numérica ordinal
job_satisfaction	Satisfacción laboral	Satisfacción laboral (0–5)	Numérica ordinal
sleep_duration	Duración del sueño	Duración del sueño (Less than 5 hours, 5–6 hours, etc.)	Categórica nominal
dietary_habits	Hábitos alimentarios	Tipo de alimentación (Healthy, Moderate, Unhealthy, Others)	Categórica nominal
degree	Programa académico / título académico	Nivel educativo alcanzado	Categórica nominal
have_you_ever_had_suicidal_thoughts_	Antecedente de pensamientos suicidas	Pensamientos suicidas (Yes/No)	Categórica nominal
work_study_hours	Horas de trabajo/estudio	Horas de trabajo/estudio por día	Numérica continua
financial_stress	Estrés financiero	Nivel de estrés financiero (0–5)	Numérica ordinal
family_history_of_mental_illness	Antecedentes familiares de enfermedad mental	Antecedentes familiares de enfermedad mental (Yes/No)	Categórica nominal
depression	Depresión	Variable objetivo: presencia de depresión (1 = Sí, 0 = No)	Variable objetivo (binaria)

Nota: A lo largo del texto, las variables podrán ser referenciadas tanto por su código original en la base de datos como por su traducción al español.

El análisis inicial del dataset mostró que, de las 18 variables, 17 no presentaban valores faltantes. La variable *financial_stress* contenía 3 valores nulos, lo que equivale al .0001% del total de las observaciones. Al ser un porcentaje muy bajo, se optó por imputar con la mediana. Por otro lado, el tipo de dato verificado mediante df.info(), confirmó que las variables numéricas estaban correctamente tipadas como *float64* o

int64 y las categóricas como *object*.

El procedimiento utilizado para cargar la base de datos y validar la variable objetivo se presenta en el Anexo B, Código B1.

3.2 Descripción estadística de las variables

3.2.1 Estadísticas descriptivas de variables numéricas

En la *Tabla 2* se resumen las principales medidas descriptivas de las variables cuantitativas del estudio (n = 27901 registros).

Tabla 2: Medidas descriptivas de variables numéricas

Variable	count	mean	sd	cv	min	25%	50%	75%	max
Age	27901	25.82	4.91	.19	18	21	25	30	59
Academic Pressure	27901	3.14	1.38	.43	0	2	3	4	5
CGPA	27901	7.66	1.47	.19	0	7	7.77	8.92	10
Study Satisfaction	27901	2.94	1.36	.46	0	2	3	4	5
Work/Study Hours	27901	7.16	3.71	.51	0	4	8	10	12
Financial Stress	27898	3.14	1.44	.45	1	2	3	4	5

Nota. Count = número de observaciones a partir de las cuales se realizó el cálculo, Sd = desviación estándar, cv = coeficiente de variación, min = mínimo, max = máximo.

La Tabla 2 presenta las medidas descriptivas de las variables numéricas consideradas relevantes para el análisis posterior. Las variables Work Pressure y Job Satisfaction no se incluyen en esta tabla debido a que presentaron valores prácticamente nulos en la muestra, con muy baja o nula variabilidad. Esta distribución es coherente con el hecho de que la base está compuesta principalmente por estudiantes, por lo que dichas variables no aportan información descriptiva sustantiva ni valor analítico para el modelado.

En cuanto a la edad, se evidenció una media de 25.82 años (DE = 4.91), con valores entre 18 y 59 años. El 50% central de los estudiantes se ubicó entre 21 y 30 años (P25 = 21; P75 = 30), lo que confirma una población mayormente joven con algunos casos de adultos, cuestión común cuando se aborda población universitaria.

Con relación a la presión académica, los resultados apuntan a una media de 3.14 (DE = 1.38) en una escala de 0 a 5. La distribución de los cuartiles (P25 = 2; P50 = 3; P75 = 4), permite afirmar que la mitad de la muestra se concentró entre niveles moderados y altos, lo cual sugiere que la presión académica constituye una experiencia relativamente frecuente entre los participantes. Entre tanto, *Work Pressure* tomó valores prácticamente nulos en toda la muestra (media ≈ 0, DE ≈ 0, cuartiles = 0), indicando que la mayoría de los participantes no reporta presión laboral, coherente con el hecho de que casi todos se identifican como estudiantes de tiempo completo.

Por otra parte, el promedio de calificaciones (*CGPA*) se ubicó en una media de 7.66 (DE = 1.47), registrado sobre una escala de 0 a 10. El 50% central se ubicó entre 6.77 y 8.92, lo que indica un rendimiento académico medio-alto, con pocos valores extremos cercanos a 0. Respecto a la variable *Study Satisfaction*,

la media fue de 2.94 (DE = 1.36), sobre una escala de 0 a 5. Si se examina la distribución, la mitad central de los estudiantes se situó entre 2 y 4 (P25 = 2; P50 = 3; P75 = 4), lo cual indica que los niveles de satisfacción oscilan entre moderados y relativamente altos. Por otro lado, *Job Satisfaction* mostró también valores prácticamente nulos (media ≈ 0, cuartiles = 0).

Finalmente, la variable *Work/Study Hours* presentó una media de 7.16 horas diarias (DE = 3.71), con un rango de 0 a 12 horas. La mediana fue de 8 horas (P25 = 4; P75 = 10), lo que indica que una proporción importante de estudiantes dedica jornadas extensas a actividades de estudio y/o trabajo. La variable *Financial Stress* arrojó una media de 3.14 (DE = 1.44) en la escala 1–5, con cuartiles 2, 3 y 4. Lo anterior, muestra que existen niveles de estrés moderados en estudiantes con baja preocupación económica como de otros que reportan niveles elevados de estrés.

3.2.2 Estadísticas descriptivas de variables categóricas

En la variable *Gender*, la distribución es relativamente equilibrada al identificar que los hombres corresponden al 55.72% de los participantes (*n* = 15547 estudiantes), mientras que las mujeres representaron el 44.28% (*n* = 12354 estudiantes). Esta proporción permite realizar comparaciones sin riesgos de sesgo fuerte por género.

La variable *City* contiene más de 50 ciudades, aunque algunas concentran la mayor proporción de estudiantes. Las más representativas son Kalyan (5.63%), Srinagar (4.92%), Hyderabad (4.80%), Vasai–Virar (4.62%) y Lucknow (4.14%). Aunque existe una larga cola de ciudades con menos del 1% de participación, esta diversidad sugiere una cobertura geográfica amplia de la población estudiantil encuestada.

Respecto a la variable *Profession*, predomina la categoría Student, que concentra 99.89% de los registros (*n* = 27870 estudiantes). El resto de las ocupaciones, como *Architect* (*n* = 8 estudiantes), *Teacher* (*n* = 6), *Digital Marketer* (*n* = 3) o *Chef* (*n* = 2), tienen una frecuencia marginal (<.05 %), confirmando que el dataset está orientado principalmente a estudiantes activos.

Los hábitos de sueño, el 29.78% de los estudiantes reportó dormir menos de cinco horas. Unos pocos, el 21.86% manifestó dormir más de ocho horas, mientras que el resto se repartieron entre quienes durmieron entre 5 y 6 horas (22.16%) y 7 a 8 horas (26.33%).

En cuanto a los hábitos alimentarios, más del 70% de la muestra presentó patrones subóptimos: el 38.98% reportó hábitos no saludables y el 35.58% hábitos moderados, mientras que solo el 27.42% indicó hábitos saludables. Lo anterior, indica que la calidad de la alimentación es un aspecto crítico del bienestar en la población analizada.

La variable Degree presentó una alta diversidad académica, con más de 20 programas distintos. Las categorías más frecuentes fueron Class 12 (21.79%), B.Ed (6.69%), B.Com (5.40%), B.Arch (5.30%) y BCA (5.14%), lo que refleja que el dataset abarca desde etapas tempranas del pregrado hasta niveles avanzados de formación.

En cuanto a las variables de salud mental se encontró que el 63.28% de los estudiantes reportó haber tenido alguna vez pensamientos suicidas (n = 17.656), el 48.4% manifestó que tienen antecedentes

familiares de enfermedad mental (n = 13.503), y la variable objetivo mostró que el 58.4% de los estudiantes de la muestra presentó depresión (n = 16.295).

3.3 Limpieza y estandarización

La etapa de limpieza partió del dataset original de 27.901 registros y 18 variables, y se desarrolló en cuatro pasos secuenciales: eliminación de variables no representativas, tratamiento de categorías no informativas, imputación de valores faltantes y eliminación de valores atípicos.

Las operaciones de limpieza, incluyendo la eliminación de variables no representativas, el tratamiento de categorías no informativas, la imputación de valores faltantes y el tratamiento de registros con CGPA = 0, se encuentran documentadas en el Anexo B, Código B1.

3.3.1 Eliminación de variables no representativas

En primer lugar, se eliminaron cinco variables que no aportaban información útil para el modelado. La variable id corresponde a un identificador único sin valor predictivo. La variable City presentó alta cardinalidad (más de 50 ciudades distintas) con una distribución muy fragmentada que dificulta su interpretación y uso en modelos. La variable Profession mostró una concentración del 99.89% en la categoría "Student", lo que la hace prácticamente constante y sin capacidad discriminante. Las variables Work Pressure y Job Satisfaction presentaron valores prácticamente nulos en la totalidad de los registros. Luego de esta eliminación, el dataset quedó con 27.901 registros y 13 variables.

3.3.2 Tratamiento de categorías no informativas

Las variables Sleep Duration y Dietary Habits presentaban una categoría denominada "Others" que representaba una proporción mínima de los datos (18 registros en Sleep Duration y 12 en Dietary Habits) y no aportaba información específica sobre los hábitos del estudiante. Estos registros fueron reemplazados por la moda de cada variable: "Less than 5 hours" para Sleep Duration y "Unhealthy" para Dietary Habits. Esta decisión permitió conservar el tamaño de la muestra sin introducir una categoría residual que pudiera generar ruido en el modelado.

3.3.3 Imputación de valores faltantes

La única variable con valores faltantes fue Financial Stress, que registró 3 valores nulos, equivalentes al .01% del total de observaciones. Al tratarse de una variable ordinal tipo Likert, se imputó con la mediana (Me = 3.0).

3.3.4 Identificación y tratamiento de valores atípicos

La variable CGPA presentó 9 registros con valor igual a cero y fueron considerados registros inválidos por lo que fueron eliminados del dataset. La variable Age mostró valores altos (superiores a los 45 años) que, aunque poco frecuentes, se consideraron plausibles dentro de una población universitaria heterogénea y se conservaron en el análisis. Las variables Academic Pressure, Study Satisfaction y Financial Stress presentaron distribuciones acordes a sus escalas ordinales y no requirieron ningún tratamiento adicional.

Finalmente, el dataset quedó conformado por 27.892 registros y 13 variables, sin valores faltantes ni registros duplicados.

3.4 Selección de variables

La definición del conjunto principal de predictores utilizado para el modelado, así como la exclusión de CGPA, Degree y Have you ever had suicidal thoughts ?, se presenta en el Anexo B, Código B1.

3.4.1 Análisis bivariado entre las variables categóricas y la presencia de síntomas depresivos

El género no mostró diferencias relevantes en la proporción de depresión entre hombres y mujeres. A pesar de esto, la variable se conservó en el modelo por su pertinencia sociodemográfica como variable de control. Se encontró que los estudiantes con menos horas de sueño mostraron una mayor proporción de casos de depresión, mientras que quienes dormían más horas mostraron una proporción menor. Esto sugiere una posible relación importante entre los hábitos de sueño y el riesgo de depresión.

Con respecto a los hábitos alimentarios, los estudiantes con hábitos no saludables presentaron la mayor proporción de depresión, seguidos por quienes reportaron hábitos moderados, y finalmente, por quienes tenían hábitos saludables, lo que sugiere una relación gradual entre la calidad de la dieta y el riesgo de depresión. Los antecedentes familiares de enfermedad mental mostraron diferencias moderadas, con una mayor proporción de depresión entre quienes los reportaron.

La variable *Degree* no evidenció diferencias sustanciales entre programas académicos y fue excluida del modelo por su alta cardinalidad y heterogeneidad conceptual. Finalmente, la variable *Have you ever had suicidal thoughts?* mostró la diferencia más marcada de todas las variables categóricas, con una proporción considerablemente mayor de depresión entre quienes reportaron pensamientos suicidas. Sin embargo, dada su alta proximidad conceptual con la variable objetivo, se decidió reservarla para un análisis complementario y excluirla del modelo principal de comparación.

3.4.2 Análisis bivariado entre las variables numéricas y la presencia de síntomas depresivos

La comparación entre las variables numéricas y la presencia de depresión se realizó a través de boxplots (Figura 1) con el fin de identificar diferencias en las medidas de tendencia central, dispersión y grupos con y sin depresión.

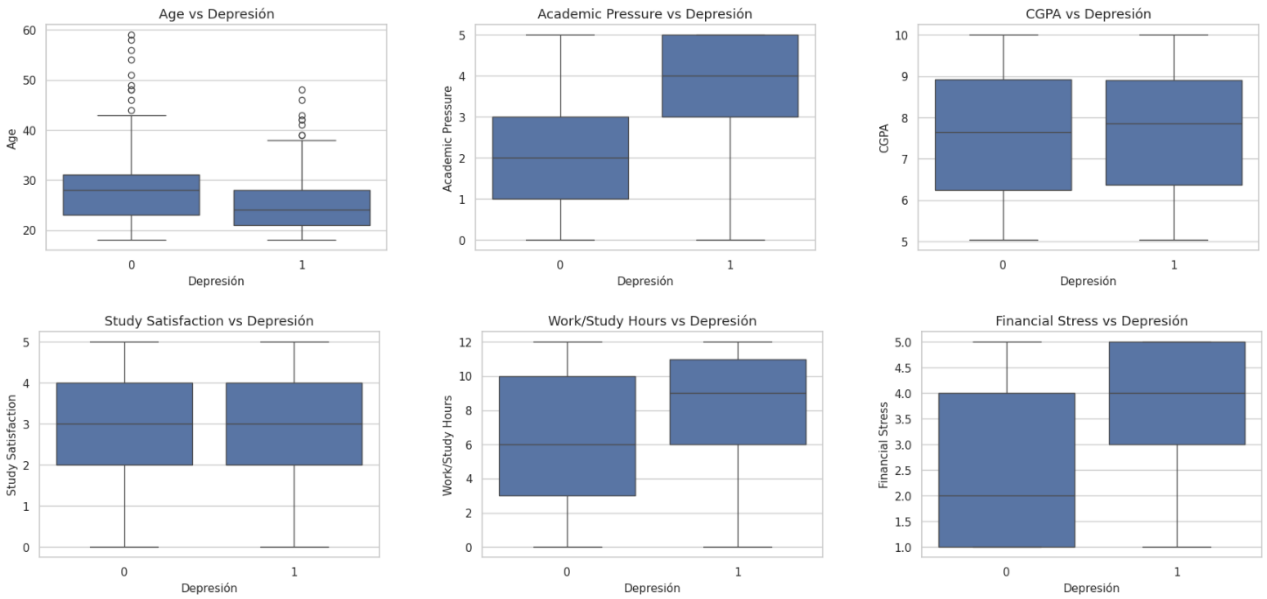

Figura 1: Variables numéricas según presencia de depresión
Nota*. 0 = sin depresión, 1 =con depresión*

Se observa que, en cuanto a la presión académica, los estudiantes con depresión presentaron niveles notablemente más altos que quienes no la presentaban, lo que refuerza su relevancia como predictor. El estrés financiero también mostró diferencias importantes en la misma dirección, con valores más altos en el grupo con depresión. Las horas de trabajo o estudio mostraron una diferencia moderada, con mayor dedicación horaria en el grupo con depresión. En cuanto a la variable edad, los estudiantes sin depresión tendieron a ser ligeramente mayores que los estudiantes con depresión. La satisfacción con los estudios mostró una diferencia moderada en dirección inversa, con niveles relativamente más altos en el grupo sin depresión. El CGPA, en cambio, no mostró diferencias apreciables entre grupos.

3.4.3 Pruebas estadísticas de asociación bivariada

Para respaldar el análisis bivariado descrito, se calcularon pruebas formales de asociación entre cada predictor y la variable objetivo. En las variables categóricas se aplicó la prueba Chi-cuadrado de independencia, y en las variables numéricas la prueba no paramétrica de Mann-Whitney U, acompañada de una verificación diagnóstica de normalidad. Estas pruebas documentan la evidencia estadística que sustenta las decisiones de selección de variables, sin modificar ninguna de ellas (ver Tabla 3).

Tabla 3: Pruebas de asociación bivariada con la variable objetivo (valores p)

Variable	Prueba	Valor p
Gender	Chi-cuadrado	.7586
Sleep Duration	Chi-cuadrado	< .001
Dietary Habits	Chi-cuadrado	< .001
Family History of Mental Illness	Chi-cuadrado	< .001
Age	Mann-Whitney U	< .001
Academic Pressure	Mann-Whitney U	< .001
Study Satisfaction	Mann-Whitney U	< .001
Work/Study Hours	Mann-Whitney U	< .001
Financial Stress	Mann-Whitney U	< .001

La mayoría de las asociaciones arrojaron valores p muy pequeños ($p < .001$). No obstante, dado el gran tamaño muestral ($n = 27.892$), casi cualquier asociación tiende a alcanzar valores p reducidos, por lo que el valor p debe interpretarse siempre junto con la magnitud del efecto (Odds Ratio o coeficiente de correlación ya reportados) y la relevancia práctica, distinguiendo la evidencia de asociación de su relevancia sustantiva. La única excepción fue la variable Gender ($p = .7586$), que no mostró una asociación apreciable con la variable objetivo; aun así, se conserva en el modelo por su pertinencia sociodemográfica como variable de control.

3.4.5 Correlaciones

Dado que varias de las variables numéricas del estudio son de naturaleza ordinal (escalas 0–5) y no cumplen el supuesto de normalidad, además del coeficiente de Pearson se calculó la matriz de correlación de Spearman, un indicador no paramétrico basado en rangos que resulta más apropiado para este tipo de datos. Los valores de Spearman difieren poco de los de Pearson, lo que refuerza la robustez de las conclusiones del análisis de correlación (ver Tabla 4).

Tabla 4: Correlaciones de Spearman entre las variables numéricas y la variable objetivo

Variable	Spearman con la clase positiva
Academic Pressure	0.47
Financial Stress	0.36
Work/Study Hours	0.20
Age	-0.23
Study Satisfaction	-0.17
CGPA	0.02

La presión académica mantiene la asociación positiva más alta con la clase positiva del dataset ($\rho = 0.47$), seguida por el estrés financiero ($\rho = 0.36$) y las horas de trabajo o estudio ($\rho = 0.20$). La edad ($\rho = -0.23$) y la satisfacción con los estudios ($\rho = -0.17$) presentan asociaciones negativas, mientras que el CGPA ($\rho = 0.02$) no muestra una asociación relevante.

Con base en el análisis exploratorio descrito, se definió el conjunto final de variables para el modelo principal. La Tabla 5 consigna las decisiones de inclusión y exclusión con sus respectivas justificaciones.

Tabla 5: Decisión sobre inclusión de variables en el modelo principal

Variable	Tipo	Decisión	Justificación
Age	Numérica	Incluir	Diferencias entre grupos y correlación negativa con depresión
Academic Pressure	Numérica	Incluir	Mayor correlación con depresión (r = .48)
Study Satisfaction	Numérica	Incluir	Relevancia conceptual y correlación negativa moderada
Work/Study Hours	Numérica	Incluir	Diferencias entre grupos y correlación positiva moderada
Financial Stress	Numérica	Incluir	Correlación relevante con depresión (r = .36)
CGPA	Numérica	Excluir	Correlación prácticamente nula con depresión (r = .02) y señales de colinealidad en el ajuste inicial
Gender	Categórica	Incluir	Variable sociodemográfica de control
Sleep Duration	Categórica	Incluir	Diferencias claras entre categorías
Dietary Habits	Categórica	Incluir	Diferencias progresivas y alta capacidad explicativa entre grupos
Family History of Mental Illness	Categórica	Incluir	Relevancia sustantiva y diferencias entre grupos
Degree	Categórica	Excluir	Alta cardinalidad, heterogeneidad conceptual y baja capacidad discriminante
Have you ever had suicidal thoughts?	Categórica	Excluir del modelo principal	Alta proximidad conceptual con la variable objetivo; se reserva para análisis complementario

3.5 Resultados Objetivo específico 1: Modelo de regresión logística

Para dar respuesta al primer objetivo específico, el análisis se llevó a cabo mediante regresión logística, un método estadístico ampliamente utilizado en ciencias de la salud precisamente porque, además de predecir, permite entender qué variables están influyendo en el resultado y en qué dirección lo hacen. En este caso, el resultado de interés es la presencia o ausencia de riesgo de depresión en los estudiantes.

A lo largo de esta sección y de las siguientes, las expresiones referidas al desenlace deben entenderse en términos de riesgo estimado y no de diagnóstico clínico. Cuando el texto alude a estudiantes clasificados según el modelo, se hace referencia a estudiantes clasificados en la clase positiva del dataset, es decir, a la probabilidad estimada de pertenecer a dicha clase. El modelo no diagnostica depresión ni determina que un estudiante la padezca; estima un riesgo orientado a la generación de alertas tempranas y al apoyo de los programas de bienestar universitario, y en ningún caso sustituye la evaluación de un profesional de salud mental. Las etiquetas “con depresión” y “sin depresión” empleadas al describir las matrices de confusión corresponden a las categorías originales del dataset utilizadas para la evaluación cuantitativa, no a un juicio clínico sobre los estudiantes.

El análisis se abordó desde dos perspectivas complementarias. La primera fue de carácter explicativo donde se buscó cuantificar la relación entre cada variable y el riesgo de depresión, con el fin de identificar cuáles factores aumentan ese riesgo y cuáles lo reducen. La segunda fue de carácter predictivo,

incorporando una técnica de regularización llamada Ridge, cuyo propósito es construir un modelo que funcione bien no solo con los datos con los que fue entrenado, sino también cuando se enfrenta a datos nuevos.

Todos los modelos fueron entrenados sobre el conjunto principal de predictores definido previamente. La construcción de X_principal, variable objetivo y así como el pipeline de preprocesamiento se presentan en el Anexo B, Código B1.

3.5.1 Diagnóstico reforzado de colinealidad y especificaciones del modelo explicativo

El diagnóstico de colinealidad mediante VIF aplica al modelo logístico explicativo clásico, cuyo propósito es la interpretación inferencial de coeficientes y odds ratios. Para este modelo explicativo se evaluó la colinealidad combinando el Factor de Inflación de la Varianza (VIF) con el número de condición de la matriz de diseño, y se contrastaron dos especificaciones: el Modelo A (especificación completa) y el Modelo B (sin la variable Age, principal foco de colinealidad) tal como se muestra en la Tabla 6.

Tabla 6: Comparación de colinealidad entre especificaciones del modelo explicativo

Especificación	VIF máximo	N.º de variables con VIF > 5	Número de condición
Modelo A (completo)	13.05	4	112.78
Modelo B (sin Age)	5.51	2	37.69

La exclusión de Age en el Modelo B reduce de forma importante la colinealidad, el VIF máximo desciende de 13.05 a 5.51, el número de variables con VIF superior a 5 pasa de 4 a 2, y el número de condición disminuye de 112.78 a 37.69. Estos resultados confirman que Age constituía el principal foco de colinealidad de la especificación clásica.

Es importante diferenciar este diagnóstico inferencial del componente predictivo del estudio. El modelo predictivo principal es la regresión logística regularizada tipo Ridge, que maneja la colinealidad mediante penalización L2 sobre los coeficientes sin eliminar variables. Por construcción, el modelo Ridge no depende de la lectura inferencial clásica del VIF, de modo que esta sección no altera la selección predictiva final: el modelo de referencia para predicción sigue siendo Logistic Ridge, evaluado más adelante.

3.5.2 Modelo logístico explicativo

El modelo convergió correctamente y obtuvo un pseudo R^2 de McFadden de .352, lo que indica que capta una porción relevante de la variabilidad asociada con el riesgo de depresión en la muestra. Los coeficientes estimados se presentan en la Tabla 7. Un coeficiente positivo significa que, a mayor valor de esa variable, mayor es la probabilidad estimada de que el estudiante presente depresión. Un coeficiente negativo indica que actúa como factor protector.

Tabla 7: Coeficientes estimados del modelo logístico explicativo

Variable	β	Error estándar	z	p-valor	IC 95% inf.	IC 95% sup.
const	-2.0855	.1205	-17.3077	<.001	-2.3217	-1.8493
Age	-.1054	.0033	-31.5517	<.001	-.1120	-.0989
Academic Pressure	.8417	.0130	64.5976	<.001	.8162	.8672
Study Satisfaction	-.2352	.0119	-19.8434	<.001	-.2584	-.2119
Work/Study Hours	.1198	.0043	27.5809	<.001	.1113	.1284
Financial Stress	.5679	.0117	48.5862	<.001	.5450	.5908
Gender_Male	.0048	.0321	.1488	.8817	-.0582	.0677
Sleep Duration_7-8 hours	.0855	.0453	1.8852	.0594	-.0034	.1743
Sleep Duration_Less than 5 hours	.3795	.0447	8.4895	<.001	.2919	.4672
Sleep Duration_More than 8 hours	-.2942	.0475	-6.1984	<.001	-.3872	-.2012
Dietary Habits_Moderate	.5063	.0395	12.8194	<.001	.4289	.5837
Dietary Habits_Unhealthy	1.0826	.0404	26.7653	<.001	1.0033	1.1619
Family History of Mental Illness_Yes	.2403	.0319	7.5410	<.001	.1778	.3027

De los 13 predictores incluidos, 11 mostraron una asociación clara con la variable objetivo respaldada por la evidencia del modelo. Las dos excepciones fueron el género masculino (p = .882) y la duración del sueño de 7 a 8 horas (p = .059), cuyo efecto no pudo diferenciarse de cero con la evidencia disponible. El caso del género llama especialmente la atención, pues su coeficiente es prácticamente nulo (β = .005), lo que sugiere que, una vez controladas las demás variables, hombres y mujeres tienen un riesgo de depresión muy similar dentro de esta muestra.

El ajuste del modelo logístico explicativo inicial y la selección de la especificación final sin Age se presentan en el Anexo B, Código B2.

3.5.3 Odds Ratios e interpretación de los resultados

Para hacer los resultados más comprensibles, se calcularon los Odds Ratios (OR), métrica que puede entenderse de manera intuitiva como una medida de cuánto cambia la probabilidad relativa de presentar depresión cuando una variable aumenta en una unidad, comparado con alguien igual en todo lo demás. Un OR de 2 indica que esa probabilidad relativa se duplica; un OR de .75 indica que se reduce en un 25%. En las variables categóricas, la interpretación se hace siempre respecto a la categoría de referencia: *Female* para género, *5–6 hours* para sueño, *Healthy* para hábitos alimentarios y *No* para antecedentes familiares.

Los resultados, ordenados de mayor a menor OR, se presentan en la Tabla 8.

Tabla 8: Odds Ratios del modelo logístico explicativo

Variable	OR	IC 95% inf.	IC 95% sup.	p-valor
Dietary Habits_Unhealthy	2.9523	2.7273	3.1959	<.001
Academic Pressure	2.3203	2.2618	2.3803	<.001
Financial Stress	1.7646	1.7246	1.8055	<.001
Dietary Habits_Moderate	1.6591	1.5355	1.7927	<.001
Sleep Duration_Less than 5 hours	1.4616	1.3390	1.5955	<.001
Family History of Mental Illness_Yes	1.2716	1.1946	1.3535	<.001
Work/Study Hours	1.1273	1.1178	1.1370	<.001
Sleep Duration_7-8 hours	1.0892	.9966	1.1904	.0594
Gender_Male	1.0048	.9435	1.0701	.8817
Age	.8999	.8940	.9058	<.001
Study Satisfaction	.7904	.7723	.8090	<.001
Sleep Duration_More than 8 hours	.7451	.6789	.8178	<.001

El predictor con mayor magnitud fue la alimentación no saludable (OR = 2.95). Comparado con un estudiante con hábitos saludables, uno con alimentación no saludable tiene casi tres veces más probabilidad relativa de presentar depresión, con todo lo demás igual. La alimentación moderada también eleva el riesgo respecto a la saludable, aunque menos (OR = 1.66), lo que sugiere que la calidad de la dieta influye de manera progresiva en el bienestar emocional del estudiante.

La presión académica fue el segundo predictor en importancia (OR = 2.32), por cada punto adicional en esa escala, la probabilidad relativa de depresión se duplica. Su magnitud refuerza la necesidad de que las instituciones universitarias atiendan la carga académica como un indicador de exigencia, así como un factor de riesgo para la salud mental. El estrés financiero ocupó el tercer lugar (OR = 1.77), lo que resulta especialmente relevante en el contexto latinoamericano, donde muchos estudiantes combinan sus estudios con responsabilidades económicas que generan una presión sostenida difícil de manejar.

Tres variables mostraron efectos protectores. La satisfacción con los estudios redujo el riesgo de manera apreciable (OR = .79), los estudiantes que se sienten más a gusto con su experiencia académica tienen menor probabilidad de presentar depresión, lo que refuerza la idea de que el bienestar en la universidad no depende solo de reducir lo negativo, sino también de promover experiencias positivas y un sentido de propósito. Dormir más de ocho horas también se asoció con menor riesgo (OR = .75), destacando el papel del sueño como factor protector. La edad, por su parte, mostró una asociación inversa continua (OR = .90 por año adicional), es decir, los estudiantes más jóvenes de la muestra son quienes concentran mayor riesgo, posiblemente porque aún están desarrollando los recursos emocionales y las estrategias de afrontamiento que suelen afianzarse con la experiencia.

La privación severa de sueño (menos de cinco horas) también se asoció con mayor riesgo (OR = 1.46), y los antecedentes familiares de enfermedad mental sumaron una vulnerabilidad adicional aunque más moderada (OR = 1.27). En conjunto, los resultados apuntan a que el riesgo de depresión en esta población está impulsado principalmente por condiciones del entorno y hábitos de vida modificables como la presión académica, el estrés financiero, la alimentación y el sueño, más que por características demográficas del

estudiante como el género o la edad. Esto tiene implicaciones directas para el diseño de programas de prevención universitaria.

No obstante, como se señaló al inicio, estos resultados deben leerse con la cautela que impone el diagnóstico de multicolinealidad, especialmente en el caso de la edad y las variables del bloque académico. La dirección general de los efectos es interpretable y coherente con la literatura, pero la precisión de algunas estimaciones puede estar afectada. Por eso, para los propósitos predictivos del proyecto, la referencia principal es el modelo Ridge que se describe a continuación.

El cálculo de los Odds Ratios, intervalos de confianza y tabla compacta final del modelo explicativo se presenta en el Anexo B, Código B2.

3.5.4 Modelo logístico con regularización Ridge

Para construir el modelo predictivo se utilizó regresión logística con regularización Ridge (L2) a través de la librería *Scikit-learn*. La elección de esta penalización responde a dos razones concretas. La primera tiene que ver con el diagnóstico de multicolinealidad presentado anteriormente, pues Ridge no elimina variables del modelo sino que reduce la magnitud de sus coeficientes, estabilizando el ajuste en presencia de correlación entre predictores sin cambiar la composición del modelo. La segunda razón es metodológica, para que la comparación posterior con el Random Forest sea válida, ambos algoritmos deben trabajar con exactamente el mismo conjunto de variables, y Ridge permite cumplir ese requisito sin modificar los predictores disponibles.

Antes de ajustar el modelo se definió el esquema de partición. El conjunto completo fue dividido de manera estratificada en 80% para entrenamiento (22313 observaciones) y 20% para prueba (5579 observaciones), asegurando que la proporción de casos positivos y negativos fuera similar en ambas partes. Esta misma partición fue conservada para el Random Forest, de modo que los dos modelos fueran evaluados sobre exactamente las mismas observaciones y cualquier diferencia en el desempeño pudiera atribuirse al método de modelado y no a la distribución de los datos.

El ajuste del modelo logístico de Ridge, la selección del hiperparámetro C, cálculo de métricas globales y evaluación por umbrales se presentan en el Anexo B, Código B3.

3.5.4.1 Selección del hiperparámetro de regularización

La regresión logística Ridge requiere definir la intensidad de la penalización mediante el parámetro C, que representa el inverso de la fuerza de regularización. En estes sentido, valores pequeños de C aplican una penalización más intensa sobre los coeficientes, mientras que valores grandes los dejan más libres. Para encontrar el valor óptimo se usó validación cruzada estratificada de cinco pliegues sobre el conjunto de entrenamiento, evaluando 13 valores de C distribuidos en escala logarítmica entre .001 y 1000. Los resultados completos de esta búsqueda se presentan en la Tabla 9.

Tabla 9: Resultados de la validación cruzada para la selección del hiperparámetro C

C	AUC entrenamiento	AUC validación cruzada	Desv. estándar validación cruzada	Posición
1.000000	.8735	.8731	.0075	1
100.000000	.8735	.8731	.0075	2
31.622777	.8735	.8731	.0075	3
316.227766	.8735	.8731	.0075	4
1000.000000	.8735	.8731	.0075	5
10.000000	.8735	.8731	.0075	6
.316228	.8735	.8731	.0075	7
3.162278	.8735	.8731	.0075	8
.100000	.8735	.8731	.0075	9
.031623	.8734	.8730	.0075	10
.010000	.8731	.8727	.0074	11
.003162	.8718	.8715	.0075	12
.001000	.8696	.8693	.0075	13

El desempeño del modelo se mantiene prácticamente idéntico a lo largo de un rango muy amplio de valores de C, desde .100 hasta 1000.000, con diferencias apenas perceptibles. Solo cuando la penalización se vuelve muy intensa (valores de C iguales a .010 o menores) el desempeño comienza a caer de manera apreciable. Esto indica que la regresión logística Ridge es bastante robusta frente a variaciones en la intensidad de la regularización, lo cual es una señal favorable de estabilidad metodológica. El valor seleccionado fue C = 1.000, con un AUC promedio en validación cruzada de .8731 y una desviación estándar entre pliegues de .0075, lo que confirma que el desempeño es consistente independientemente de cómo se distribuyan los datos en cada pliegue.

3.5.4.2 Coeficientes regularizados

La Tabla 10 presenta los coeficientes del modelo Ridge, ordenados de mayor a menor valor absoluto. A diferencia del modelo explicativo, estos coeficientes no se interpretan como estimaciones inferenciales directas sino como parámetros ajustados bajo penalización, cuyo propósito es reflejar la dirección y la importancia relativa de cada predictor dentro de una especificación orientada a la predicción [29].

Tabla 10: Coeficientes regularizados del modelo logístico Ridge

Variable	Coeficiente (β)
num__Academic Pressure	1.1646
cat__Dietary Habits_Unhealthy	1.1114
num__Financial Stress	.8209
cat__Dietary Habits_Moderate	.5337
num__Age	-.5129
num__Work/Study Hours	.4331
cat__Sleep Duration_Less than 5 hours	.3778
num__Study Satisfaction	-.3295
cat__Sleep Duration_More than 8 hours	-.2907
cat__Family History of Mental Illness_Yes	.2570
cat__Sleep Duration_7-8 hours	.1070
cat__Gender_Male	.0220

La presión académica y los hábitos alimentarios no saludables encabezan la lista con los coeficientes más altos, seguidos por el estrés financiero. La edad, la satisfacción con los estudios y dormir más de ocho horas conservan sus coeficientes negativos, actuando como factores de protección relativa dentro del modelo. El género masculino vuelve a mostrar una contribución prácticamente nula (β = .022), confirmando que no tiene un papel relevante en la estimación del riesgo bajo esta especificación.

3.5.4.3 Desempeño global en el conjunto de prueba

Una vez ajustado el modelo, se generaron las probabilidades predichas para el conjunto de prueba y se evaluó el desempeño con tres métricas complementarias. El AUC-ROC mide la capacidad del modelo para distinguir entre casos positivos y negativos a lo largo de todos los posibles umbrales de clasificación. La Precisión Promedio (AP) resume el comportamiento de la curva Precisión-Recall y resulta especialmente informativa cuando el interés principal está en la identificación de casos positivos; y el Brier Score mide qué tan bien calibradas están las probabilidades predichas, es decir, qué tan fielmente reflejan el riesgo real observado en los datos. Los resultados se presentan en la Tabla 11.

Tabla 11: Métricas globales del modelo Logistic Ridge en el conjunto de prueba

Métrica	Valor
AUC entrenamiento	.8735
AUC prueba	.8666
Precisión Promedio (AP)	.8924
Brier Score	.1438
Brecha AUC (entrenamiento – prueba)	.0069

La Figura 2 muestra la curva ROC del modelo tanto en entrenamiento como en prueba. El eje horizontal representa la tasa de falsas alarmas (1 – especificidad) y el eje vertical la tasa de detección de casos positivos (sensibilidad), para todos los posibles umbrales de clasificación. Una curva que se acerca a la esquina superior izquierda del gráfico indica buena capacidad para detectar casos reales mientras mantiene baja la tasa de falsas alarmas; la diagonal punteada representa el desempeño de un clasificador al azar, que sirve como línea de referencia mínima.

Como se observa en la figura, ambas curvas, entrenamiento y prueba, se alejan claramente de esa diagonal y se ubican en la zona superior izquierda del gráfico, lo que refleja un desempeño sólido en los dos conjuntos. La cercanía entre las dos curvas es, además, evidencia directa de que el modelo no se ajustó en exceso a los datos de entrenamiento: una brecha grande entre ambas sería señal de sobreajuste, pero aquí la diferencia en AUC es de apenas .0069, lo que confirma que el desempeño es estable cuando el modelo se enfrenta a datos que no ha visto antes. También se puede identificar sobre la curva el punto correspondiente al umbral de *Youden* (.6087), que marca el punto de máximo balance entre sensibilidad y especificidad [30].

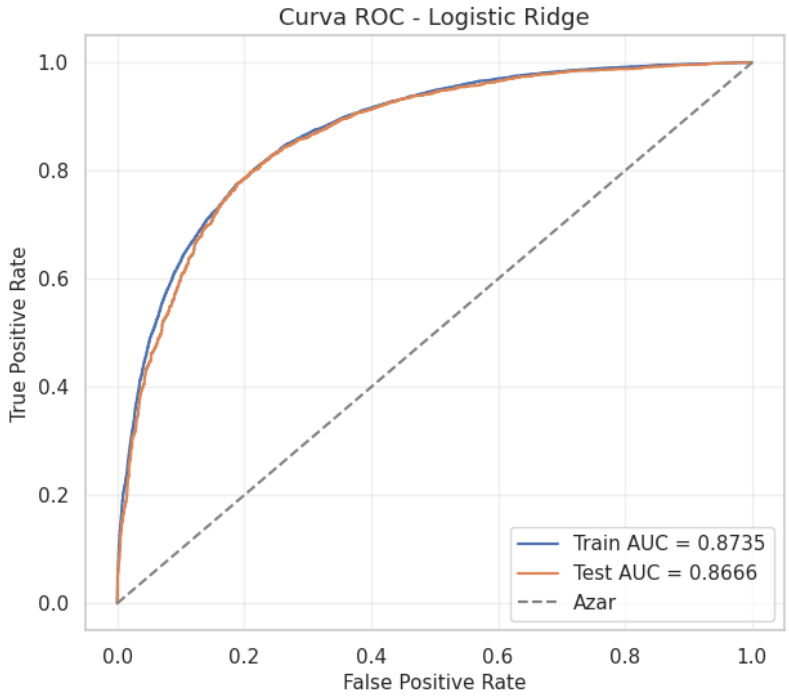

Figura 2: Curva ROC del modelo Logistic Ridge

El Brier Score de .1438 complementa la lectura del AUC mostrando que las probabilidades producidas por el modelo son útiles para clasificar y también son confiables como estimaciones del nivel de riesgo individual de cada estudiante. Esto se puede ver en la Figura 3, que muestra la curva de calibración donde el eje horizontal representa las probabilidades predichas por el modelo y el eje vertical la proporción real de casos positivos observados en cada grupo de estudiantes. La diagonal punteada representa una calibración perfecta, es decir, el escenario ideal donde la probabilidad predicha coincide exactamente con la frecuencia observada.

La curva del modelo Logistic Ridge sigue de cerca esa diagonal a lo largo de todo el rango de probabilidades, tanto en los valores bajos, donde el modelo estima poco riesgo y efectivamente hay pocos casos de depresión, como en los valores altos, donde la proporción real de casos se acerca de manera consistente a la probabilidad predicha.

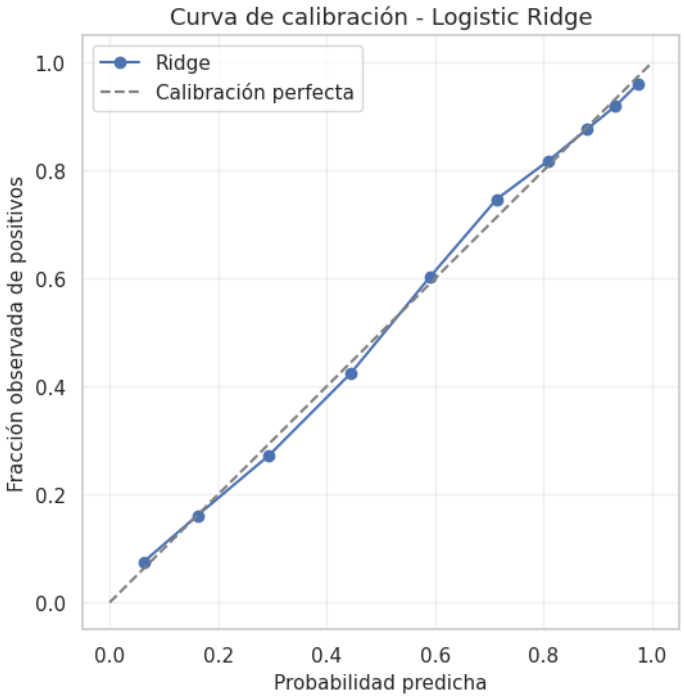

Figura 3: Curva de calibración del modelo Logistic Ridge

3.5.4.4 Evaluación por umbrales de clasificación

Para clasificar a los estudiantes como casos en riesgo o sin riesgo se evaluaron tres criterios de corte, cuyos resultados se presentan en la Tabla 12 y se ilustran en la Figura 8 mediante las matrices de confusión correspondientes a cada umbral. La elección del umbral tiene consecuencias prácticas concretas: un falso negativo es un estudiante en riesgo que el modelo no detecta, privándolo de una posible intervención oportuna; un falso positivo es un estudiante sin riesgo que queda señalado como caso, generando una alerta innecesaria que puede traducirse en una carga adicional sobre los servicios de bienestar institucional.

Tabla 12: Desempeño del modelo Logistic Ridge por umbral de clasificación

Umbral	Sensibilidad	Especificidad	Precisión	F1-score	TN	FP	FN	TP
.50	.8519	.7206	.8116	.8312	1666	646	484	2783
.20	.9666	.3936	.6925	.8070	910	1402	109	3158
.6087 (*Youden*)	.7796	.8054	.8498	.8132	1862	450	720	2547

Las matrices de confusión de la Figura 4 permiten visualizar de manera directa cómo se distribuyen los aciertos y los errores del modelo en cada umbral. Cada matriz organiza las predicciones en cuatro celdas: los verdaderos negativos (estudiantes sin depresión correctamente identificados), los falsos positivos (estudiantes sin depresión señalados como en riesgo), los falsos negativos (estudiantes con depresión no detectados) y los verdaderos positivos (estudiantes con depresión correctamente identificados). Comparar las tres matrices permite apreciar visualmente el efecto de ajustar el umbral: a medida que el umbral baja de .6087 a .50 y luego a .20, los verdaderos positivos aumentan pero también lo hacen los falsos positivos,

mientras que los falsos negativos disminuyen progresivamente.

Con el umbral estándar de .50, el modelo encontró un equilibrio razonable, detectando correctamente el 85.2% de los estudiantes con depresión mientras clasificaba correctamente como negativos al 72.1% de quienes no la presentaban. El umbral de .20 es el más sensible de los tres puesto que deja sin detectar a apenas 109 estudiantes en riesgo sobre los 3.267 presentes en el conjunto de prueba, una tasa de falsos negativos muy baja que lo hace atractivo para un tamizaje masivo. Sin embargo, esto viene acompañado de 1.402 falsas alarmas, lo que representaría una carga operativa importante para los equipos de bienestar si no existe una etapa de seguimiento posterior.

El umbral de *Youden* (.6087) fue el que ofreció el mejor balance entre ambos tipos de error, con una sensibilidad de .7796 y una especificidad de .8054, y la precisión más alta de los tres (.8498). Vale aclarar que la elección definitiva del umbral en un contexto institucional real va más allá del análisis: depende de la capacidad de respuesta de los servicios disponibles, del costo de una intervención innecesaria y, sobre todo, de las consecuencias de dejar sin atención a un estudiante que sí lo necesita.

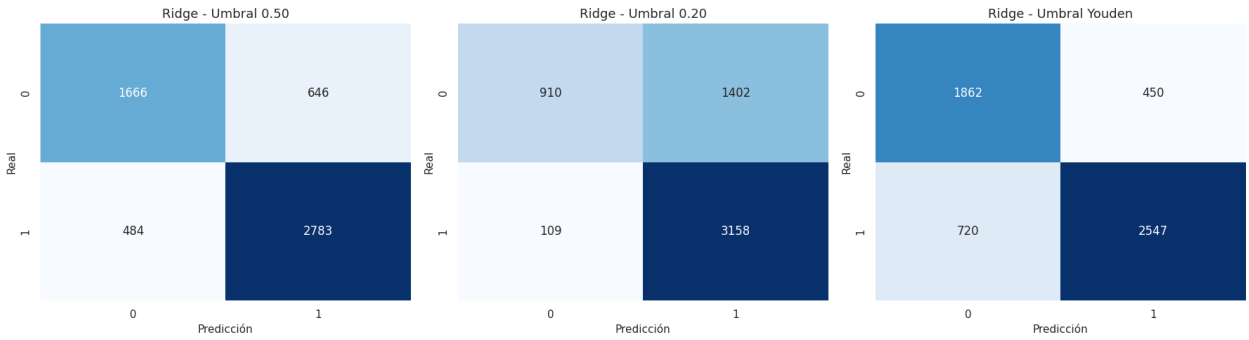

Figura 4: Matrices de confusión del modelo Logistic Ridge para los tres umbrales de clasificación
Nota: ***0 = sin depresión, 1 =con depresión***

3.6 Resultados Objetivo específico 2: Random Forest Classifier

El Random Forest Classifier (RFC) es un método de aprendizaje automático que construye simultáneamente un gran número de árboles de decisión, cada uno entrenado sobre una muestra aleatoria de los datos y con un subconjunto aleatorio de variables disponibles en cada división [24]. La predicción final se obtiene combinando los resultados de todos los árboles mediante votación, lo que hace al modelo robusto frente al sobreajuste y capaz de capturar relaciones complejas entre variables que un modelo lineal como la regresión logística no puede detectar [31]. A diferencia de Ridge, el RFC no asume ninguna forma funcional para la relación entre los predictores y la variable objetivo, lo que lo convierte en una alternativa complementaria y metodológicamente valiosa para contrastar los hallazgos del modelo logístico.

Para esta etapa se utilizó exactamente la misma partición de datos empleada en el modelo Ridge (80% train, 20% test), con estratificación para preservar la proporción de casos positivos y negativos en ambos conjuntos. Esto garantiza que cualquier diferencia en el desempeño entre los dos modelos sea atribuible al método y no a las condiciones de evaluación.

3.6.1 Selección de hiperparámetros

El RFC requiere definir varios hiperparámetros antes de ser ajustado: el número de árboles a construir (n_estimators), la profundidad máxima de cada árbol (max_depth), el número mínimo de observaciones necesarias para dividir un nodo (min_samples_split) y para que un nodo sea hoja (min_samples_leaf), y la proporción de variables consideradas en cada división (max_features). Para encontrar la combinación óptima se realizó una búsqueda exhaustiva sobre una grilla de valores usando la misma validación cruzada estratificada de cinco pliegues empleada con el modelo Ridge, evaluando el desempeño mediante el AUC.

Los diez mejores resultados de esta búsqueda se presentan en la Tabla 13. Al igual que ocurrió con el parámetro C en el modelo Ridge, varios conjuntos de hiperparámetros produjeron desempeños muy similares, lo que indica que el RFC también es razonablemente estable frente a variaciones en su configuración.

Tabla 13: Mejores configuraciones de hiperparámetros del Random Forest en validación cruzada

n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features	AUC entrenamiento	AUC validación cruzada
500	10	2	5	sqrt	.9110	.8703
500	10	10	5	sqrt	.9110	.8703
500	10	10	1	sqrt	.9154	.8702
300	10	10	1	sqrt	.9152	.8701
300	10	10	5	sqrt	.9108	.8701
300	10	2	5	sqrt	.9108	.8701
500	10	2	1	sqrt	.9227	.8700
300	10	2	1	sqrt	.9226	.8697
500	20	2	5	sqrt	.9384	.8689
500	20	10	5	sqrt	.9384	.8689

La configuración seleccionada fue: 500 árboles, profundidad máxima de 10, mínimo de 2 observaciones para dividir un nodo, mínimo de 5 observaciones en hojas terminales y selección de variables mediante raíz cuadrada. Esta configuración alcanzó un AUC promedio en validación cruzada de .8703. Vale la pena notar que las configuraciones con mayor profundidad (max_depth = 20 o sin límite) obtuvieron AUC de entrenamiento considerablemente más altos, pero su desempeño en validación cruzada fue menor, lo que es una señal clara de sobreajuste: esos árboles más profundos aprenden mejor los datos de entrenamiento, pero generalizan peor.

El pipeline del Random Forest Classifier y la búsqueda de hiperparámetros mediante validación cruzada se presentan en el Anexo B, Código B4.

3.6.2 Importancia relativa de los predictores

Una de las ventajas más relevantes del RFC es que permite cuantificar cuánto contribuye cada variable a las decisiones del modelo. Esta medida, conocida como importancia de los predictores, se calcula a partir de cuánto reduce cada variable la impureza promedio en los nodos donde es utilizada para dividir, ponderado por el número de observaciones que pasan por esos nodos. La Tabla 14 presenta los resultados agrupados por variable original.

Tabla 14: Importancia relativa de los predictores en el Random Forest Classifier

Variable	Importancia relativa
Academic Pressure	.4099
Financial Stress	.2163
Age	.1170
Work/Study Hours	.0882
Dietary Habits	.0643
Study Satisfaction	.0520
Sleep Duration	.0300
Family History of Mental Illness	.0117
Gender	.0105

La presión académica emergió como el predictor más importante por amplio margen, concentrando el 41% de la capacidad clasificatoria del modelo. Esto significa que de todas las variables analizadas, la presión académica es la que más contribuye a distinguir entre estudiantes con y sin riesgo de depresión dentro del RFC. El estrés financiero ocupó el segundo lugar con el 21.6%, seguido por la edad con el 11.7%. Juntas, estas tres variables explican aproximadamente el 74% de la importancia total del modelo, lo que sugiere que el riesgo de depresión en esta muestra está fuertemente concentrado en un conjunto reducido de factores contextuales.

Las horas de trabajo o estudio (8.8%) y los hábitos alimentarios (6.4%) también aportaron información predictiva relevante, aunque en menor medida. La satisfacción con los estudios (5.2%) completó el grupo de variables con contribución apreciable. En el extremo opuesto, los antecedentes familiares de enfermedad mental y el género fueron los predictores con menor peso dentro del modelo, con importancias de apenas 1.2% y 1.1% respectivamente, lo que refuerza el hallazgo del modelo logístico de que el riesgo no se distribuye de manera diferencial entre hombres y mujeres una vez se consideran los demás factores.

Este ranking de importancia es coherente en su estructura con los coeficientes del modelo Ridge, lo que da consistencia a los hallazgos del proyecto en su conjunto: independientemente del enfoque metodológico empleado, lineal o no lineal, los mismos factores emergen como los más relevantes para comprender el riesgo de depresión en la población universitaria analizada.

El cálculo de la importancia relativa de los predictores se presenta en el Anexo B, Código B4.

3.6.3 Análisis de patrones no lineales

Una de las capacidades distintivas del RFC frente a la regresión logística es su habilidad para capturar relaciones no lineales entre los predictores y la variable objetivo. Para explorar este aspecto, se construyeron gráficos de dependencia parcial para las tres variables de mayor importancia: presión académica, estrés financiero y edad. Estos gráficos muestran cómo cambia, en promedio, la probabilidad estimada de depresión cuando varía cada predictor de manera aislada, manteniendo constantes los demás factores del modelo. Los resultados se presentan en la Figura 5.

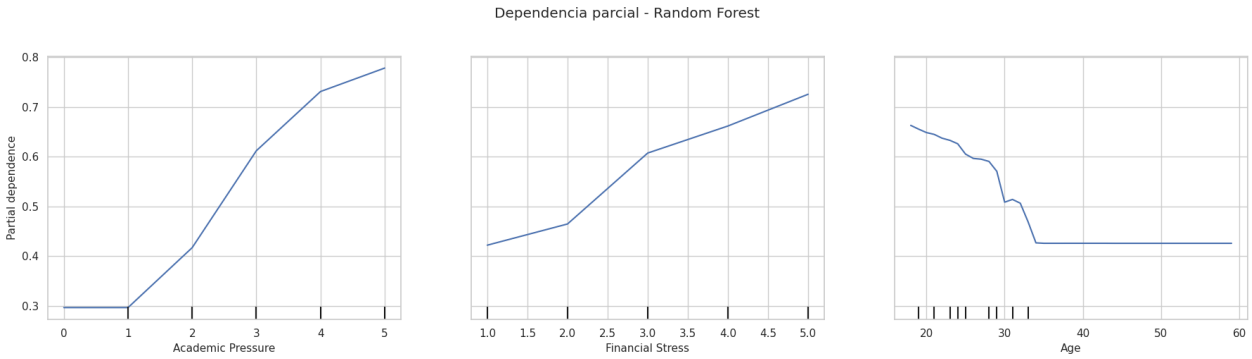

Figura 5: Gráficos de dependencia parcial del Random Forest Classifier para las tres variables de mayor importancia

En el caso de la presión académica, el patrón que muestra el RFC no es lineal. La probabilidad estimada de depresión se mantiene prácticamente estable en los niveles más bajos de la escala y luego se eleva de manera marcada a partir de valores intermedios, especialmente en el rango entre los niveles 2 y 4. Este comportamiento sugiere que no cualquier nivel de presión académica incrementa el riesgo de la misma manera sino que, existe una especie de umbral a partir del cual el efecto se intensifica considerablemente. En otras palabras, pasar de una presión baja a una moderada tiene un impacto diferente al de pasar de una moderada a una alta.

El estrés financiero mostró un patrón similar pero con algunas particularidades. La relación es positiva y creciente a lo largo de toda la escala, pero la pendiente no es uniforme. El incremento en la probabilidad de depresión es más moderado en los niveles bajos del estrés y se acelera progresivamente en los niveles intermedios y altos. Este comportamiento sugiere que el estrés financiero opera de manera acumulativa, con un efecto que se va intensificando a medida que la situación económica del estudiante se vuelve más apremiante.

La edad presentó el patrón más llamativo de los tres. La relación es inversa (a mayor edad, menor probabilidad estimada de depresión) pero no es constante en todo el rango. La caída más pronunciada en el riesgo estimado se concentra en la transición entre los veinte y los treinta años, mientras que en edades superiores la curva tiende a estabilizarse, con una reducción del riesgo más gradual. Esto sugiere que los primeros años de la vida universitaria, cuando los estudiantes son más jóvenes, representan el período de mayor vulnerabilidad dentro de la muestra analizada, y que los recursos de afrontamiento que se van desarrollando con la experiencia y la madurez tienen un efecto protector más marcado precisamente en ese tramo de la vida.

En conjunto, estos resultados muestran que el RFC identifica matices en las relaciones entre variables y riesgo de depresión que una regresión logística, por su naturaleza lineal, no puede capturar con la misma precisión. Aunque esto no fue suficiente para que el RFC superara a Ridge en las métricas globales de desempeño, sí aporta una lectura complementaria valiosa desde el punto de vista sustantivo; las relaciones entre los principales factores de riesgo y la depresión no son uniformes a lo largo de sus escalas, y las intervenciones institucionales podrían ser más efectivas si se orientan específicamente hacia los rangos donde el incremento del riesgo es más pronunciado.

3.6.4 Desempeño global en el conjunto de prueba

Los resultados del desempeño global del RFC se presentan en la Tabla 15.

Tabla 15: Métricas globales del modelo Random Forest Classifier en el conjunto de prueba

Métrica	**Valor**
AUC entrenamiento	.9067
AUC prueba	.8648
Precisión Promedio (AP)	.8901
Brier Score	.1460
Brecha AUC (entrenamiento – prueba)	.0419

El AUC de prueba de .8648 indica una buena capacidad para distinguir entre estudiantes con y sin riesgo de depresión, comparable al obtenido por el modelo Ridge. Sin embargo, a diferencia de lo que ocurrió con Ridge, el RFC muestra una brecha de .0419 entre el AUC de entrenamiento y el de prueba, lo que indica una tendencia al sobreajuste moderado. El modelo aprende con mayor profundidad los patrones del conjunto de entrenamiento, de ahí su AUC de .9067 en esa fase, pero no logra trasladar completamente esa ganancia a datos nuevos. Esta es una característica habitual de los métodos de ensamble con alta capacidad de aprendizaje, y es precisamente la razón por la que se impuso un límite de profundidad máxima de 10 niveles durante la búsqueda de hiperparámetros. Sin ese límite, la brecha habría sido aún mayor.

La Figura 6 muestra la curva ROC del RFC en entrenamiento y prueba. A diferencia de lo observado en el modelo Ridge, donde ambas curvas eran muy cercanas entre sí, en el RFC se aprecia una separación más visible entre la curva de entrenamiento y la de prueba, lo que ilustra gráficamente esa mayor tendencia al sobreajuste. Aun así, la curva de prueba se mantiene claramente por encima de la diagonal de referencia, confirmando que el modelo conserva una capacidad discriminante sólida cuando se enfrenta a datos nuevos.

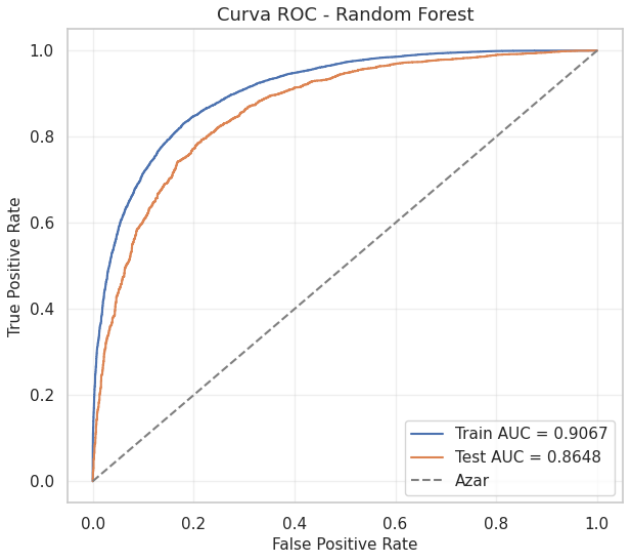

Figura 6: Curva ROC del modelo Random Forest Classifier

El Brier Score de .1460 indica que las probabilidades producidas por el RFC también son razonablemente confiables, aunque ligeramente menos calibradas que las del modelo Ridge (.1438). Esto se puede ver en la Figura 7, que muestra la curva de calibración del RFC. A diferencia del modelo logístico, cuya curva seguía de cerca la diagonal de referencia a lo largo de todo el rango, el RFC tiende a comprimir sus probabilidades hacia el centro. En los deciles bajos subestima el riesgo real y en los deciles altos lo subestima también, aunque de manera menos pronunciada. Este comportamiento es característico de los modelos de ensamble basados en votación, que por su naturaleza producen probabilidades con menor variabilidad que los modelos paramétricos, lo que puede limitar su uso como estimadores precisos del riesgo individual.

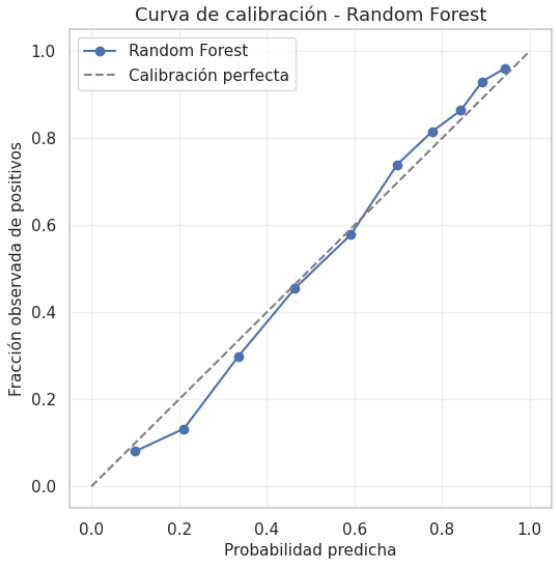

Figura 7: Curva de calibración del modelo Random Forest Classifier

El cálculo de las métricas globales del Random Forest Classifier se presenta en el Anexo B, Código B4.

3.6.5 Evaluación por umbrales de clasificación

Al igual que con el modelo Ridge, se evaluaron tres umbrales de clasificación para el RFC. El umbral de *Youden* se calculó sobre el conjunto de entrenamiento y resultó ser .5808. Los resultados se presentan en la Tabla 13 y las matrices de confusión correspondientes en la Tabla 16.

Tabla 16: Desempeño del modelo Random Forest Classifier por umbral de clasificación

Umbral	Sensibilidad	Especificidad	Precisión	F1-score	TN	FP	FN	TP
.50	.8555	.7059	.8043	.8291	1632	680	472	2795
.20	.9786	.3123	.6679	.7939	722	1590	70	3197
.5808 (*Youden*)	.7974	.7751	.8336	.8151	1792	520	662	2605

Nota. TP = verdaderos positivos; TN = verdaderos negativos; FP = falsos positivos; FN = falsos negativos.

El comportamiento del RFC por umbral sigue un patrón similar al del modelo Ridge. Con el umbral estándar de .50, detectó correctamente el 85.6% de los estudiantes con depresión con una especificidad del 70.6%. El umbral de .20 llevó la detección hasta el 97.9%, dejando sin identificar apenas 70 casos positivos en todo el conjunto de prueba, aunque generando 1.590 falsas alarmas. El umbral de *Youden* (.5808) ofreció el mejor equilibrio, con una sensibilidad de .7974 y una especificidad de .7751.

Comparando los umbrales de *Youden* de ambos modelos, el RFC muestra una especificidad ligeramente menor que el Ridge (.7751 frente a .8054), mientras que su sensibilidad es marginalmente mayor (.7974 frente a .7796). Las matrices de confusión de la Figura 8*: Matrices de confusión del modelo Random Forest Classifier para los tres umbrales de clasificación* permiten visualizar estas diferencias directamente, mostrando cómo se distribuyen los aciertos y errores del RFC en cada punto de corte.

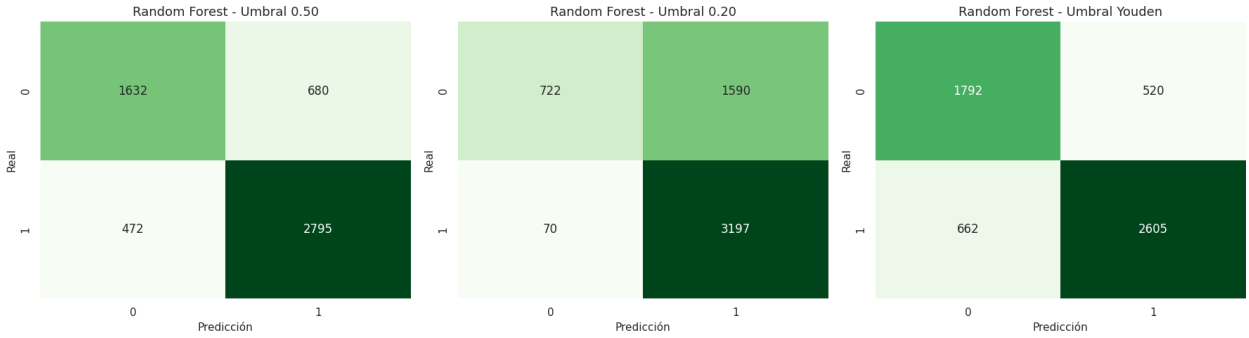

Figura 8: Matrices de confusión del modelo Random Forest Classifier para los tres umbrales de clasificación

Nota: *0 = sin depresión, 1 =con depresión*

La evaluación del Random Forest Classifier bajo los umbrales 0.50, 0.20 y Youden se presenta en el Anexo B, Código B4.

3.7 Resultados Objetivo específico 3: Comparación entre modelos

Una vez evaluados por separado los dos modelos, el paso siguiente fue compararlos de manera directa. Para que esta comparación fuera válida desde el punto de vista metodológico, era necesario que ambos algoritmos hubieran sido entrenados y evaluados bajo exactamente las mismas condiciones, lo implicó emplear el mismo conjunto de variables predictoras, la misma partición entre entrenamiento y prueba, y el mismo criterio de optimización durante la búsqueda de hiperparámetros. Todas estas condiciones se cumplieron, por lo que cualquier diferencia observada en el desempeño puede atribuirse al comportamiento propio de cada método y no a factores externos al modelado.

3.7.1 Comparación de métricas globales

La Tabla 17 resume las métricas globales de ambos modelos en los conjuntos de entrenamiento y prueba.

Tabla 17: Comparación de métricas globales entre modelos

Modelo	AUC entrenamiento	AUC prueba	Precisión Promedio (AP)	Brier Score
Logistic Ridge	.8735	.8666	.8924	.1438
Random Forest	.9067	.8648	.8901	.1460

En cuanto a los resultados, a pesar de que el RFC alcanzó un AUC de entrenamiento considerablemente más alto que el modelo *Ridge* (.9067 frente a .8735), esta ventaja desapareció casi por completo cuando ambos modelos fueron evaluados sobre el conjunto de prueba: Ridge obtuvo .8666 y RFC .8648, una diferencia de apenas .0018 puntos. En otras palabras, el RFC aprendió más profundamente los datos de entrenamiento, pero no logró trasladar esa ventaja a datos nuevos, mientras que Ridge mantuvo un desempeño muy estable entre entrenamiento y prueba.

Esta lectura se refuerza con la Precisión Promedio y el Brier Score. En ambos casos Ridge obtuvo valores marginalmente mejores: AP de .8924 frente a .8901, y Brier Score de .1438 frente a .1460. Aunque las diferencias son pequeñas, apuntan en la misma dirección, puesto que el modelo *Ridge* produce probabilidades ligeramente más precisas y mejor calibradas que el RFC cuando se enfrenta a datos que no ha visto antes.

El cálculo de la tabla comparativa de métricas globales entre Logistic Ridge y Random Forest Classifier se presenta en el Anexo B, Código B5.

3.7.2 Validación cruzada repetida y estabilidad de las métricas

Para evaluar la estabilidad de las métricas ante cambios en las particiones de entrenamiento y validación, se aplicó una validación cruzada repetida tipo Repeated Stratified K-Fold con 5 pliegues y 5 repeticiones (25 evaluaciones por modelo) sobre el conjunto de entrenamiento. Este procedimiento complementa, pero no reemplaza, la evaluación final sobre el conjunto de prueba. No corresponde a una validación anidada, ya que utiliza los hiperparámetros previamente seleccionados sin reoptimizarlos dentro de cada pliegue; su función es estimar la dispersión e incertidumbre de las métricas. La Tabla 18 reporta la media y la desviación estándar de cada métrica.

Tabla 18: Validación cruzada repetida (5×5) — media y desviación estándar por métrica

Métrica	Logistic Ridge (media)	Logistic Ridge (DE)	Random Forest (media)	Random Forest (DE)
AUC	0.8732	0.0057	0.8702	0.0061
Average Precision	0.9007	0.0049	0.8987	0.0053
Brier Score	0.1409	0.0033	0.1437	0.0031
Accuracy	0.7922	0.0051	0.7925	0.0063
Sensibilidad	0.7940	0.0161	0.7978	0.0097
Especificidad	0.7897	0.0207	0.7849	0.0157
Precisión	0.8424	0.0109	0.8399	0.0091
F1-score	0.8173	0.0057	0.8182	0.0055

La validación repetida confirma que Logistic Ridge y Random Forest tienen desempeños muy cercanos. Ridge mantiene una ligera ventaja en AUC, Average Precision y Brier Score (mejor calibración), mientras que Random Forest resulta competitivo en sensibilidad y F1-score. Las desviaciones estándar son pequeñas en ambos modelos, lo que evidencia estabilidad ante el cambio de particiones. Estas diferencias deben leerse como diferencias prácticas y no como una superioridad estadística absoluta; la selección de Ridge se sustenta en el mejor equilibrio entre desempeño, estabilidad, calibración e interpretabilidad.

3.7.3 Intervalos de confianza Bootstrap de las métricas en prueba.

Para comunicar la incertidumbre de las métricas globales sobre el conjunto de prueba, se estimaron intervalos de confianza del 95% mediante bootstrap (2.000 remuestreos). El procedimiento remuestrea únicamente las predicciones ya generadas, sin reentrenar los modelos. Los resultados se presentan en la Tabla 19.

Tabla 19: Intervalos de confianza bootstrap (IC95%) de las métricas en el conjunto de prueba

Modelo	AUC [IC95%]	AP [IC95%]	Brier [IC95%]
Logistic Ridge	0.8666 [0.8562, 0.8761]	0.8924 [0.8817, 0.9022]	0.1438 [0.1384, 0.1497]
Random Forest	0.8648 [0.8548, 0.8740]	0.8901 [0.8790, 0.9005]	0.1460 [0.1412, 0.1511]

Los intervalos de confianza de ambos modelos se solapan ampliamente en las tres métricas, lo que respalda que tienen un desempeño predictivo muy similar. Ridge conserva una ligera ventaja en calibración probabilística, reflejada en su menor Brier Score.

3.7.4 Comparación pareada de la diferencia de AUC

Para dar sustento estadístico a la afirmación de que la diferencia de desempeño entre ambos modelos es mínima, se calculó un intervalo de confianza del 95% por bootstrap pareado para la diferencia de AUC (Ridge menos Random Forest) sobre el mismo conjunto de prueba (ver Tabla 20).

Tabla 20: Diferencia de AUC entre modelos (bootstrap pareado, IC95%)

Diferencia AUC (Ridge − RF)	IC95% inferior	IC95% superior	¿Incluye cero?
0.0019	-0.0010	0.0046	Sí

La diferencia de AUC es muy pequeña (0.0019) y su intervalo de confianza incluye el cero. Por tanto, no debe afirmarse que Ridge sea estadísticamente superior a Random Forest en capacidad de discriminación; la lectura correcta es que no hay evidencia de una diferencia de desempeño entre los modelos. En consecuencia, la elección de Ridge no se justifica por superioridad estadística, sino por su equilibrio entre desempeño, menor complejidad, mejor calibración e interpretabilidad.

3.7.5 Comparación de métricas globales

La Tabla 21 pone en perspectiva la brecha entre el desempeño en entrenamiento y en prueba para cada modelo, que es uno de los indicadores más directos de la estabilidad de un modelo predictivo.

Tabla 21: Comparación de la brecha AUC entre entrenamiento y prueba

Modelo	AUC entrenamiento	AUC prueba	Brecha AUC
Logistic Ridge	.8735	.8666	.0069
Random Forest	.9067	.8648	.0419

La brecha del RFC (.0419) es seis veces más grande que la del modelo Ridge (.0069). Esto indica que el RFC presenta una tendencia al sobreajuste moderado en tanto el modelo se ajusta con mayor intensidad a los datos de entrenamiento, lo que le permite rendir mejor en esa etapa, pero esa ganancia no se transfiere de manera equivalente al conjunto de prueba. El Ridge, en cambio, muestra una generalización muy estable, con un comportamiento prácticamente idéntico en ambos conjuntos. Esta diferencia en estabilidad es relevante en un contexto aplicado, donde el modelo será eventualmente usado con datos de estudiantes nuevos que el modelo nunca ha procesado antes.

El cálculo de la brecha AUC entre entrenamiento y prueba para ambos modelos se presenta en el Anexo B, Código B5.

3.7.6 Comparación por umbrales de clasificación

Más allá de las métricas globales, la Tabla 22 compara el comportamiento operativo de ambos modelos bajo los tres umbrales de clasificación evaluados a lo largo del estudio.

Tabla 22: Comparación de métricas por umbral de clasificación

Modelo	Umbral	Sensibilidad	Especificidad	Precisión	F1-score	FN	FP
Logistic Ridge	.50	.8519	.7206	.8116	.8312	484	646
Random Forest	.50	.8555	.7059	.8043	.8291	472	680
Logistic Ridge	.20	.9666	.3936	.6925	.8070	109	1402
Random Forest	.20	.9786	.3123	.6679	.7939	70	1590
Logistic Ridge	.6087 (*Youden*)	.7796	.8054	.8498	.8132	720	450
Random Forest	.5808 (*Youden*)	.7974	.7751	.8336	.8151	662	520

Con el umbral estándar de .50, los dos modelos se comportan de manera muy similar. El RFC detecta marginalmente más casos positivos (sensibilidad .8555 frente a .8519), pero lo hace a costa de generar más falsas alarmas (680 frente a 646). Ridge, por su parte, clasifica correctamente como negativos a más estudiantes que no tienen depresión (especificidad .7206 frente a .7059).

Con el umbral orientado a sensibilidad de 0.20, el RFC detecta más casos positivos (97.86% frente a 96.66%), dejando sin identificar apenas 70 estudiantes en riesgo frente a los 109 del modelo Ridge. Sin embargo, esta ganancia viene acompañada de 1.590 falsas alarmas frente a las 1.402 del Ridge, representando una carga operativa adicional considerable.

Con el umbral de *Youden*, donde cada modelo usa el punto de corte óptimo calculado sobre sus propios datos de entrenamiento, el RFC muestra una sensibilidad algo mayor (.7974 frente a .7796), mientras que Ridge mantiene una especificidad superior (.8054 frente a .7751). Esta diferencia refleja una tendencia general que se mantiene en los tres umbrales, la cual consiste en que el RFC tiende a detectar más casos positivos, pero generando también más falsas alarmas, mientras que Ridge ofrece un balance ligeramente más favorable entre ambos tipos de error.

La comparación de sensibilidad, especificidad, precisión, F1-score y matrices de confusión bajo los umbrales evaluados se presenta en el Anexo B, Código B5.

3.7.7 Comparación visual de curvas ROC

La Figura 9 presenta las curvas ROC de ambos modelos en el conjunto de prueba sobre un mismo gráfico, lo que permite apreciar visualmente su capacidad discriminante comparada. Como se puede observar, las dos curvas se superponen de manera notable a lo largo de casi todo su recorrido, lo que confirma que el desempeño de ambos modelos en prueba es prácticamente equivalente. No existe un modelo que domine claramente al otro a lo largo de todos los umbrales posibles, puesto que, en algunos tramos Ridge se posiciona ligeramente por encima, y en otros el RFC lo supera marginalmente, sin que ninguno de los dos establezca una ventaja consistente y clara sobre el otro.

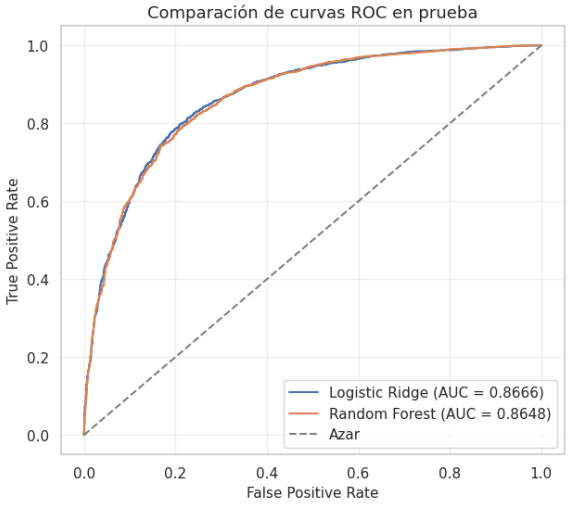

Figura 9: Comparación de curvas ROC en el conjunto de prueba

La construcción de las curvas ROC comparativas en el conjunto de prueba se presenta en el Anexo B, Código B5.

3.7.8 Selección del modelo de referencia

Logistic Ridge fue seleccionado como modelo de referencia porque alcanzó un desempeño predictivo comparable al de Random Forest, mostró mejor calibración probabilística, menor complejidad y mayor interpretabilidad. En un contexto de salud mental universitaria, donde el modelo debe apoyar alertas tempranas y no reemplazar una evaluación clínica, la interpretabilidad y la estabilidad de las probabilidades estimadas son criterios especialmente relevantes.

Random Forest se conserva como modelo comparativo flexible, útil para capturar patrones no lineales e importancia relativa de los predictores, pero no mostró una mejora suficientemente amplia para desplazar a Logistic Ridge como modelo de referencia.

4 CONCLUSIONES Y TRABAJOS FUTUROS

4.1 CONCLUSIONES

Este proyecto aplicado desarrolló e implementó dos modelos de aprendizaje automático para estimar el riesgo de depresión en estudiantes universitarios que incluyeron una regresión logística con regularización Ridge y un modelo de ensamble con la técnica Random Forest Classifier. Ambos modelos fueron entrenados y evaluados sobre el Student Depression Dataset, utilizando un conjunto común de variables sociodemográficas, académicas y de hábitos de vida. A partir de los resultados obtenidos, es posible establecer conclusiones en tres niveles: metodológico, sustantivo y aplicado.

El proyecto demostró que es posible construir modelos predictivos del riesgo de depresión en población universitaria con un desempeño sólido a partir de variables disponibles en encuestas de bienestar estudiantil. Ambos modelos alcanzaron valores de AUC superiores a .86 en el conjunto de prueba, lo que indica una buena capacidad para distinguir entre estudiantes con y sin riesgo de depresión. No obstante, los dos modelos no se comportaron de la misma manera frente al sobreajuste, ya que la regresión logística Ridge mostró una brecha de apenas .0069 entre su desempeño en entrenamiento y prueba, mientras que el RFC presentó una brecha de .0419, lo que indica que el bosque aprendió más intensamente los datos de entrenamiento sin trasladar esa ganancia de manera equivalente a datos nuevos. Esta diferencia en la estabilidad de generalización fue determinante para seleccionar Ridge como modelo de referencia, dado que en un contexto aplicado el modelo será usado continuamente con datos de estudiantes que no participaron en el entrenamiento.

La comparación entre los dos enfoques también mostró que la mayor complejidad del RFC no se tradujo en una mejora real del desempeño. En métricas como la Precisión Promedio (AP de .8924 vs. .8901) y el Brier Score (.1438 vs. .1460), el modelo Ridge obtuvo valores marginalmente mejores. Al evaluar la diferencia de AUC entre ambos modelos mediante bootstrap pareado, el intervalo de confianza del 95% incluye el cero, por lo que la diferencia de capacidad de discriminación se encuentra dentro de la variabilidad esperada del conjunto de prueba y no puede atribuirse una ventaja clara a ninguno de los dos. En consecuencia, la selección de Ridge no se sustenta en una superioridad de discriminación, sino en su mejor calibración probabilística, su mayor estabilidad de generalización y su interpretabilidad. Este hallazgo es consistente con el principio de parsimonia que orienta la selección de modelos en ciencia de datos, según el cual, cuando dos modelos ofrecen un desempeño comparable, se prefiere el más sencillo, el que generaliza mejor y el que permite comunicar sus resultados de manera más transparente a equipos no especializados.

Los resultados fueron consistentes entre los dos modelos y señalaron un conjunto reducido de factores como los principales determinantes del riesgo de depresión en la muestra analizada. La presión académica fue el predictor más relevante en ambos enfoques, al obtener un OR de 2.32 en el modelo logístico explicativo, mientras que en el RFC concentró el 41% de la importancia total del modelo. Los hábitos alimentarios no saludables presentaron la mayor magnitud de asociación en el modelo logístico (OR = 2.95), lo que indica que los estudiantes con alimentación deficiente presentan cerca de tres veces la oportunidad (odds) de depresión en comparación con quienes tienen hábitos saludables. El estrés financiero completó el grupo de predictores más relevantes, con un OR de 1.77 en el modelo explicativo y una importancia relativa del 21.6% en el RFC.

En el otro extremo, la satisfacción con los estudios (OR = .79), dormir más de ocho horas (OR = .75) y la edad (OR = .90 por año adicional) actuaron como factores protectores. El género, en cambio, no mostró una contribución relevante en ninguno de los dos modelos, lo que sugiere que en este conjunto de datos el riesgo de depresión se distribuye de manera similar entre hombres y mujeres una vez se controlan los factores académicos, económicos y de hábitos de vida. El análisis de dependencia parcial del RFC añadió una dimensión adicional a esta lectura, desde el cual se apreció que las relaciones entre presión académica, estrés financiero y depresión no son uniformes a lo largo de sus escalas, sino que se intensifican a partir de ciertos rangos intermedios, lo que tiene implicaciones concretas para el diseño de intervenciones preventivas.

Es preciso indicar que los resultados de este proyecto tienen un alcance metodológico y no clínico. Los modelos desarrollados no realizan diagnósticos de depresión ni reemplazan la evaluación de profesionales de salud mental, ya que su función es identificar patrones de riesgo a partir de variables estructurales y de hábitos de vida, con el fin de alertar a las instituciones sobre posibles casos que merecen atención psicosocial especializada. La decisión sobre qué tipo de intervención aplicar en cada caso corresponde a los equipos de bienestar universitario, quienes cuentan con el criterio profesional y el contexto institucional necesarios para hacer esa valoración de manera responsable y ética.

Estas conclusiones deben leerse dentro de las restricciones del dataset público de Kaggle empleado. Ello en virtud de que el estudio empleó una base pública de acceso libre, sin embargo, no existe conocimiento expreso de instrumento de medición, la forma de recolección de la información ni la validez clínica de la etiqueta de la variable objetivo. Además, los datos provienen de un contexto distinto al colombiano, lo que limita la transferencia directa de los hallazgos. Por ello, el modelo estima un riesgo dentro del marco de los datos analizados y su aplicación a una población local exigiría una validación previa con información propia. En este sentido, la principal utilidad del análisis aquí desarrollado radica en explorar modelos de estudio a partir de los cuales se pueden realizar aplicaciones a un fenómeno educativo y social de alta relevancia. Es necesario formular nuevos estudios que permitan contrastar si el comportamiento de los modelos es estable y aplicable a datos reales.

De cumplirse con ese criterio, los resultados de este trabajo pueden servir de un insumo valioso para contribuir al campo de la salud mental en contextos educativos a partir de la ciencia de datos. El proceso metodológico desarrollado, desde la exploración y limpieza de datos hasta la comparación y aplicación de modelos predictivos, puede servir como base replicable para que instituciones de educación superior diseñen sistemas de alerta temprana adaptados a sus propias poblaciones estudiantiles, complementando así sus programas de bienestar con herramientas de análisis basadas en evidencia.

4.1 LIMITACIONES Y TRABAJOS FUTUROS

El presente estudio presenta algunas limitaciones que deben considerarse al interpretar los resultados y al proyectar futuras aplicaciones del modelo. Una de ellas se relaciona con la selección del umbral de clasificación. En esta investigación, el umbral óptimo se definió mediante la maximización del índice de *Youden*, calculado como $J = sensibilidad + especificidad - 1$, sobre el conjunto de entrenamiento. Este procedimiento permitió obtener un umbral de .6087, con una sensibilidad de .78, una especificidad de .81 y un índice de *Youden* de 0.59. Posteriormente, dicho umbral fue aplicado al subconjunto de prueba y a la muestra completa para generar las clasificaciones finales.

Aunque esta estrategia es común en estudios exploratorios y permitió obtener resultados estables, como lo evidencia el AUC de 0.8648 en el conjunto de prueba, la estimación del umbral a partir de una única partición de los datos puede limitar la capacidad de generalización del modelo ante nuevas muestras. En consecuencia, para trabajos futuros se recomienda calcular el umbral mediante validación cruzada estratificada dentro del conjunto de entrenamiento. Este procedimiento permitiría estimar el índice de *Youden* en diferentes pliegues, promediar los resultados y aplicar un umbral más robusto al conjunto de prueba.

Otra limitación corresponde al origen de la base de datos utilizada. El estudio se apoya en un dataset público de Kaggle con información limitada sobre el instrumento de medición utilizado. La variable de desenlace se trata como una etiqueta binaria propia del dataset y no como un diagnóstico clínico confirmado, por lo que su validez clínica no puede establecerse con la información disponible.

Adicionalmente, en este estudio no se realizó validación externa en población colombiana, de manera que los patrones identificados deben extrapolarse con cautela a contextos distintos al del dataset. Los modelos estiman un riesgo dentro del marco de los datos analizados y la herramienta resultante debe usarse como apoyo a la decisión, nunca como un mecanismo de decisión automática que reemplace la valoración profesional.

Además, la base de datos empleada contiene un conjunto limitado de variables asociadas a la depresión estudiantil. Aunque se incluyeron factores relevantes como presión académica, estrés financiero, satisfacción con el estudio, hábitos de sueño, antecedentes familiares y pensamientos asociados al estado emocional, existen otras dimensiones que podrían fortalecer el análisis. En futuras investigaciones sería pertinente incorporar variables relacionadas con el acceso a servicios de salud mental, redes de apoyo, integración social en la institución, consumo de sustancias psicoactivas, participación en actividades de bienestar e historial de eventos vitales estresantes. La inclusión de estas variables permitiría construir un perfil de riesgo más amplio y contextualizado.

De igual manera, el modelo desarrollado corresponde a un análisis de corte transversal, ya que estima el riesgo en un momento específico. Esta característica limita la posibilidad de observar cambios en el estado del estudiante a lo largo del tiempo. Como línea futura, sería valioso desarrollar modelos longitudinales que permitan monitorear la evolución del riesgo en una misma persona durante diferentes periodos académicos. Este enfoque facilitaría la identificación de trayectorias de aumento, disminución o permanencia del riesgo, aportando información útil para la detección temprana y el acompañamiento institucional.

También se identifica como oportunidad futura el desarrollo de una interfaz de consulta dirigida a los equipos de bienestar universitario. Esta herramienta permitiría ingresar información de los estudiantes y obtener estimaciones de riesgo en tiempo real, facilitando la priorización de casos y el diseño de rutas de acompañamiento. Para avanzar en esta dirección, sería necesario definir protocolos éticos sobre confidencialidad, consentimiento informado, protección de datos personales, interpretación de resultados y uso responsable de las predicciones. De esta forma, el modelo podría funcionar como apoyo a la toma de decisiones, sin reemplazar la valoración profesional ni los procesos institucionales de atención.

Finalmente, futuros estudios podrían incorporar técnicas de aprendizaje no supervisado, como el análisis de conglomerados, con el propósito de identificar subgrupos de estudiantes con perfiles de riesgo diferenciados. Este tipo de análisis permitiría reconocer patrones asociados, por ejemplo, a presión académica, estrés financiero, hábitos de sueño o dificultades de integración social. A partir de esta segmentación, las instituciones podrían diseñar estrategias de prevención y acompañamiento más ajustadas a las características de cada grupo.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] E. S. Ariza Mora, D. Y. Almeyda Ortiz, y S. F. Garcia Rueda, «Depresión en estudiantes universitarios en carreras de la salud En Bucaramanga 2020», dic. 2020, Accedido: 31 de mayo de 2025. [En línea]. Disponible en: https://hdl.handle.net/20.500.12494/34561

[2] M. de Souza Martins, M. X. Figueroa Angel, S. Toro Arévalo, y L. Gonçalves Junior, «Modelo de predicción multivariado entre los factores psicológicos y los estilos de vida con la percepción de calidad de vida de estudiantes universitarios», *Retos Nuevas Tend. En Educ. Física Deporte Recreación*, n.º 53, pp. 427-436, 2024.

[3] M. del C. T. Saldívar, J. A. R. E. Mota, A. R. Cabral, M. E. Reyes-Robles, y J. P. Macías, «Prevalencia de depresión en estudiantes de algunas carreras del Centro de Ciencias de la Salud de la Universidad Autónoma de Aguascalientes», *Lux Médica*, vol. 9, n.º 26, Art. n.º 26, ene. 2014, doi: 10.33064/26lm2014853.

[4] «(PDF) Estrés académico y cansancio emocional en estudiantes universitarios: Un estudio transversal», *ResearchGate*, dic. 2024, doi: 10.47307/GMC.2024.132.1.6.

[5] «Depressive disorder (depression)». Accedido: 26 de mayo de 2025. [En línea]. Disponible en: https://www.who.int/news-room/fact-sheets/detail/depression

[6] Y. Zhang *et al.*, «Global, regional and national burdens of major depression disorders and its attributable risk factors in adolescents and young adults aged 10-24 years from 1990 to 2021», *BMC Psychiatry*, vol. 25, n.º 1, p. 399, abr. 2025, doi: 10.1186/s12888-025-06772-w.

[7] A. P. M. Amaltinga y J. F. Mbinta, «Factors associated with depression among young people globally: a narrative review», *Int. J. Community Med. Public Health*, vol. 7, n.º 9, pp. 3711-3721, ago. 2020, doi: 10.18203/2394-6040.ijcmph20203949.

[8] «(PDF) The Impact of Depressive and Anxiety Symptoms on Academic Achievement among Undergraduate University Students: A Systematic Review», *ResearchGate*, ene. 2025, doi: 10.21926/obm.neurobiol.2404261.

[9] S. T. T. Morales, G. del P. V. Quinchalef, J. A. A. Vera, S. I. M. Muñoz, y K. M. W. Contreras, «Niveles de depresión, ansiedad, estrés y su relación con el rendimiento académico en estudiantes universitarios», *Investig. En Educ. Médica*, vol. 9, n.º 36, Art. n.º 36, oct. 2020, doi: 10.22201/fm.20075057e.2020.36.20229.

[10] «apoyo-a-las-necesidades-sociales-emocionales-conductuales-y-de-salud-mental-de-ninos-y-estudiantes.pdf». Accedido: 30 de mayo de 2025. [En línea]. Disponible en: https://www.ed.gov/sites/ed/files/documents/students/apoyo-a-las-necesidades-sociales-emocionales-conductuales-y-de-salud-mental-de-ninos-y-estudiantes.pdf

[11] «(PDF) Salud Mental en Estudiantes de Educación Superior: Un Desafío Post-pandemia»,

ResearchGate. Accedido: 13 de mayo de 2025. [En línea]. Disponible en: https://www.researchgate.net/publication/380398530_Salud_Mental_en_Estudiantes_de_Educacion_Superior_Un_Desafio_Post-pandemia

[12] «(PDF) Machine learning predictive models to guide prevention and intervention allocation for anxiety and depressive disorders among college students». Accedido: 14 de mayo de 2025. [En línea]. Disponible en: https://www.researchgate.net/publication/385101751_Machine_learning_predictive_models_to_guide_prevention_and_intervention_allocation_for_anxiety_and_depressive_disorders_among_college_students

[13] O. Iparraguirre-Villanueva, C. Paulino-Moreno, A. Epifanía-Huerta, y C. Torres-Ceclén, «Machine Learning Models to Classify and Predict Depression in College Students», *Int. J. Interact. Mob. Technol. IJIM*, vol. 18, n.º 14, Art. n.º 14, ago. 2024, doi: 10.3991/ijim.v18i14.48669.

[14] «Mental health». Accedido: 30 de mayo de 2025. [En línea]. Disponible en: https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response

[15] «Prevalence and associated factors of depression and anxiety symptoms among college students: a systematic review and meta-analysis | Request PDF», *ResearchGate*, doi: 10.1111/jcpp.13606.

[16] «Cross-Sectional Analysis of Colombian University Students' Perceptions of Mental Health during the COVID-19 Pandemic: Repercussions on Academic Achievement». Accedido: 30 de mayo de 2025. [En línea]. Disponible en: https://www.mdpi.com/2227-9032/11/14/2024

[17] L. Sotaquirá *et al.*, «Social Capital and Lifestyle Impacts on Mental Health in University Students in Colombia: An Observational Study», *Front. Public Health*, vol. 10, p. 840292, 2022, doi: 10.3389/fpubh.2022.840292.

[18] «The CES-D Scale - Lenore Sawyer Radloff, 1977». Accedido: 30 de mayo de 2025. [En línea]. Disponible en: https://journals.sagepub.com/doi/abs/10.1177/014662167700100306

[19] S. V. Alpi y A. O. Bechara, «Variables asociadas a la ansiedad-depresión en estudiantes universitarios», *Univ. Psychol.*, vol. 19, 2020, doi: 10.11144/Javeriana.upsy19.vaad.

[20] «Depresión». Accedido: 30 de mayo de 2025. [En línea]. Disponible en: https://www.who.int/es/news-room/fact-sheets/detail/depression

[21] A. B. R. Shatte, D. M. Hutchinson, y S. J. Teague, «Machine learning in mental health: a scoping review of methods and applications», *Psychol. Med.*, vol. 49, n.º 9, pp. 1426-1448, jul. 2019, doi: 10.1017/S0033291719000151.

[22] M. I. Jordan y T. M. Mitchell, «Machine learning: Trends, perspectives, and prospects», *Science*, vol. 349, n.º 6245, pp. 255-260, jul. 2015, doi: 10.1126/science.aaa8415.

[23] «Introduction to the Logistic Regression Model», en *Applied Logistic Regression*, John Wiley & Sons,

Ltd, 2013, pp. 1-33. doi: 10.1002/9781118548387.ch1.

[24] L. Breiman, «Random Forests», *Mach. Learn.*, vol. 45, n.º 1, pp. 5-32, oct. 2001, doi: 10.1023/A:1010933404324.

[25] «3.4. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.6.1 documentation». [En línea]. Disponible en: https://scikit-learn.org/stable/modules/model_evaluation.html

[26] «(PDF) Enhancing Mental Health Disorder Detection: A Hybrid Classifier System Using Soft Voting and Ensemble Methods», *ResearchGate*, abr. 2025, doi: 10.36948/ijfmr.2025.v07i01.35963.

[27] S. Góngora Alonso, «Aplicación de técnicas de machine learning en la predicción de hospitalizaciones y reingresos de pacientes con esquizofrenia en Castilla y León», http://purl.org/dc/dcmitype/Text, Universidad de Valladolid, 2023. Accedido: 31 de mayo de 2025. [En línea]. Disponible en: https://portalcienciaytecnologia.jcyl.es/documentos/658098bfd726106b34d7b9bd

[28] F. A. Amú Ruiz, D. A. Gonzalez Bustamante, K. Ortiz González, y K. M. Sandoval, «Árbol de clasificación para la identificación de síntomas asociados a la depresión en estudiantes de una universidad pública», *Retos Nuevas Tend. En Educ. Física Deporte Recreación*, n.º 52, pp. 104-114, 2024.

[29] C. J. Greenwood *et al.*, «A comparison of penalised regression methods for informing the selection of predictive markers», *PLOS ONE*, vol. 15, n.º 11, p. e0242730, nov. 2020, doi: 10.1371/journal.pone.0242730.

[30] «IBM SPSS Statistics». Accedido: 11 de mayo de 2026. [En línea]. Disponible en: https://www.ibm.com/docs/en/spss-statistics/30.0.0?topic=schemes-area-under-curve&utm_source=chatgpt.com

[31] «An Introduction to Statistical Learning», An Introduction to Statistical Learning. Accedido: 11 de mayo de 2026. [En línea]. Disponible en: https://www.statlearning.com

6 ANEXOS

Anexo A. Índice de fragmentos de código por objetivo específico

En este anexo se presenta la relación entre los fragmentos principales del código y las secciones del documento en las que se reportan los resultados. No se incluye la totalidad del cuaderno con el fin de evitar redundancia con las tablas y figuras del documento; se conservan únicamente los bloques necesarios para garantizar trazabilidad metodológica entre la preparación de datos, el modelado, la comparación de algoritmos y la clasificación final.

Código	Fragmento de código	Qué debe contener	Sección relacionada del documento
B1	Preparación general de datos, selección de variables y análisis bivariado	Importación de librerías, semilla aleatoria, carga del dataset, validación de la variable objetivo, limpieza, selección de variables, creación de X_principal y y, pipeline de preprocesamiento, pruebas de asociación bivariada y correlación de Spearman	2. Metodología; 3.1; 3.2; 3.3; 3.4.1 a 3.4.3
B2	Regresión logística explicativa y diagnóstico de colinealidad	Matriz de diseño, codificación dummy, categorías de referencia, diagnóstico VIF, número de condición, comparación entre especificaciones con y sin Age, ajuste con statsmodels.Logit, coeficientes, p-valores, intervalos de confianza y Odds Ratios	3.5.1; 3.5.2; 3.5.3
B3	Regresión logística Ridge	Pipeline Ridge, partición train/test, validación cruzada, selección de C, métricas globales, curva ROC, calibración, evaluación por umbrales y probabilidades estimadas	3.5.4
B4	Random Forest Classifier	Pipeline, GridSearchCV, hiperparámetros, mejor configuración, importancia de predictores, métricas de desempeño, curva ROC, calibración y evaluación por umbrales	3.6.1 a 3.6.5
B5	Comparación entre modelos y análisis de robustez	Tabla comparativa Ridge vs. Random Forest, AUC train/test, Average Precision, Brier Score, brecha AUC, comparación por umbrales, curva ROC conjunta, validación cruzada repetida, intervalos de confianza bootstrap y diferencia de AUC mediante bootstrap pareado	3.7.1 a 3.7.5

Anexo B. Fragmentos principales del código

Código B 1. Preparación general de datos

El Código B1 presenta el fragmento del cuaderno utilizado para la preparación general de los datos. Incluye la configuración inicial, carga del dataset, validación de la variable objetivo, limpieza, selección de variables, creación de X_principal y y, construcción del pipeline de preprocesamiento y análisis bivariado mediante pruebas estadísticas y correlación de Spearman. Se omiten bloques auxiliares de exportación, visualización y salidas intermedias que no son necesarios para la trazabilidad metodológica.

```
# ============================================================================
# 1. CONFIGURACIÓN DEL ENTORNO Y ESTRUCTURA DEL PROYECTO
# ============================================================================

# -------------------------
# IMPORTS BASE
# -------------------------
from pathlib import Path
import os
import warnings
import random
from datetime import datetime

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

warnings.filterwarnings("ignore")

# -------------------------
# CONFIGURACIÓN GLOBAL
# -------------------------
RANDOM_STATE = 42
np.random.seed(RANDOM_STATE)
random.seed(RANDOM_STATE)

sns.set_theme(style="whitegrid", context="notebook")

plt.rcParams["figure.figsize"] = (8, 5)
plt.rcParams["axes.titlesize"] = 13
plt.rcParams["axes.labelsize"] = 11

print("Configuración general establecida.")


# ============================================================================
# 1.1 DEFINICIÓN DE RUTAS DEL PROYECTO
# ============================================================================

USAR_DRIVE = False

try:
    from google.colab import drive
```

```
    drive.mount('/content/drive')
    USAR_DRIVE = True
    print("Google Drive montado correctamente.")
except:
    print("Ejecución en entorno local.")

# Ruta base
if USAR_DRIVE:
    BASE_PROYECTO = Path("/content/drive/MyDrive/Proyecto_Tesis_Depresion")
else:
    BASE_PROYECTO = Path.cwd() / "Proyecto_Tesis_Depresion"

# Carpetas principales
RUTAS = {
    "base_datos": BASE_PROYECTO / "01_Base_de_datos",
    "resultados": BASE_PROYECTO / "03_Resultados_generados",
    "documento": BASE_PROYECTO / "04_Documento_de_tesis",
    "registro": BASE_PROYECTO / "05_Registro_metodologico"
}

# Subcarpetas de resultados
SUBCARPETAS_RESULTADOS = {
    "ini": "01_Configuracion_y_control",
    "lim": "02_Limpieza_y_preparacion",
    "eda": "03_Analisis_descriptivo",
    "mlg": "04_Modelo_regresion_logistica",
    "rfc": "05_Modelo_random_forest",
    "com": "06_Comparacion_de_modelos",
    "cla": "07_Clasificacion_final",
    "doc": "08_Anexos_para_documento",
}

# Crear carpetas
for ruta in RUTAS.values():
    ruta.mkdir(parents=True, exist_ok=True)

for clave, nombre in SUBCARPETAS_RESULTADOS.items():
    (RUTAS["resultados"] / nombre).mkdir(parents=True, exist_ok=True)

print("Estructura de carpetas lista.")

# ============================================================================
# 1.2 SISTEMA DE EXPORTACIÓN Y REGISTRO
# ============================================================================

REGISTRO_EXPORTS = []

def registrar_exportacion(nombre_archivo, seccion, descripcion, ruta):
    REGISTRO_EXPORTS.append({
        "fecha": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "seccion": seccion,
        "nombre_archivo": nombre_archivo,
        "descripcion": descripcion,
        "ruta": str(ruta)
    })

def exportar_csv(df, nombre_archivo, seccion, descripcion):
    carpeta = RUTAS["resultados"] / SUBCARPETAS_RESULTADOS[seccion]
    ruta = carpeta / nombre_archivo
```

```
    df.to_csv(ruta, index=False)
    registrar_exportacion(nombre_archivo, seccion, descripcion, ruta)

def exportar_excel(df, nombre_archivo, seccion, descripcion):
    carpeta = RUTAS["resultados"] / SUBCARPETAS_RESULTADOS[seccion]
    ruta = carpeta / nombre_archivo
    df.to_excel(ruta, index=False)
    registrar_exportacion(nombre_archivo, seccion, descripcion, ruta)

def exportar_grafica(nombre_archivo, seccion, descripcion):
    carpeta = RUTAS["resultados"] / SUBCARPETAS_RESULTADOS[seccion]
    ruta = carpeta / nombre_archivo
    plt.savefig(ruta, bbox_inches="tight")
    registrar_exportacion(nombre_archivo, seccion, descripcion, ruta)

import re

def limpiar_nombre_archivo(texto):
    """
    Convierte un texto en un nombre de archivo seguro:
    - pasa a minúsculas
    - reemplaza espacios por guiones bajos
    - elimina caracteres problemáticos
    """
    texto = texto.lower().strip()
    texto = texto.replace(" ", "_")
    texto = texto.replace("/", "_")
    texto = texto.replace("\\", "_")
    texto = texto.replace("?", "")
    texto = re.sub(r"[^a-z0-9_áéíóúüñ]", "", texto)
    return texto

# ============================================================================
# 1.3 EXPORTACIÓN DEL ÍNDICE GENERAL
# ============================================================================

def exportar_indice_general():
    if len(REGISTRO_EXPORTS) == 0:
        print("No hay archivos registrados aún.")
        return

    df_indice = pd.DataFrame(REGISTRO_EXPORTS)

    ruta_excel = RUTAS["resultados"] / "01_Configuracion_y_control" / "INI-
indice_general_de_archivos_generados.xlsx"
    ruta_csv = RUTAS["resultados"] / "01_Configuracion_y_control" / "INI-
indice_general_de_archivos_generados.csv"

    df_indice.to_excel(ruta_excel, index=False)
    df_indice.to_csv(ruta_csv, index=False, encoding="utf-8-sig")

    print("Índice general exportado correctamente en Excel y CSV.")

# ============================================================================
# 2. PARÁMETROS DEL ESTUDIO
# ============================================================================

# -------------------------
# IDENTIFICACIÓN GENERAL DEL PROYECTO
```

```
# -------------------------
NOMBRE_PROYECTO = "Predicción del riesgo de depresión en estudiantes universitarios"
VERSION_CUADERNO = "Versión 5"
FECHA_EJECUCION = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# -------------------------
# ARCHIVO PRINCIPAL DE DATOS
# -------------------------
NOMBRE_ARCHIVO_DATOS = "Student Depression Dataset.csv"
RUTA_ARCHIVO_DATOS = RUTAS["base_datos"] / NOMBRE_ARCHIVO_DATOS

# -------------------------
# VARIABLE OBJETIVO
# -------------------------
VARIABLE_OBJETIVO = "Depression"

# -------------------------
# UMBRALES DE CLASIFICACIÓN A EVALUAR
# 0.50: criterio estándar de clasificación binaria.
# 0.20: umbral orientado a sensibilidad, útil para explorar escenarios
#       de mayor detección de casos positivos.
# El umbral basado en Youden se calculará posteriormente a partir de la ROC.
# -------------------------
UMBRAL_ESTANDAR = 0.50
UMBRAL_SENSIBLE = 0.20
CALCULAR_UMBRAL_YOUDEN = True

# -------------------------
# NOMBRES OFICIALES DE LOS MODELOS DEL ESTUDIO
# -------------------------
MODELO_LOGISTICO = "Regresión logística"
MODELO_RANDOM_FOREST = "Random Forest Classifier"

# -------------------------
# PREFIJOS OFICIALES PARA EXPORTACIONES
# -------------------------
PREFIJOS_EXPORTACION = {
    "ini": "INI",
    "lim": "LIM",
    "eda": "EDA",
    "mlg": "MLG",
    "rfc": "RFC",
    "com": "COM",
    "cla": "CLA",
    "doc": "DOC"
}

# -------------------------
# RESUMEN DE PARÁMETROS
# -------------------------
print("Proyecto:", NOMBRE_PROYECTO)
print("Versión del cuaderno:", VERSION_CUADERNO)
print("Fecha de ejecución:", FECHA_EJECUCION)
print("Archivo esperado:", RUTA_ARCHIVO_DATOS)
print("Variable objetivo esperada:", VARIABLE_OBJETIVO)
print("Umbral estándar:", UMBRAL_ESTANDAR)
print("Umbral orientado a sensibilidad:", UMBRAL_SENSIBLE)
print("Calcular umbral de Youden:", CALCULAR_UMBRAL_YOUDEN)
print("Modelos del estudio:", MODELO_LOGISTICO, "y", MODELO_RANDOM_FOREST)
```

```
# ============================================================================
# 2.1 TABLA DE CONTROL DE PARÁMETROS DEL ESTUDIO
# ============================================================================

tabla_parametros = pd.DataFrame({
    "parametro": [
        "nombre_proyecto",
        "version_cuaderno",
        "fecha_ejecucion",
        "archivo_principal_de_datos",
        "variable_objetivo_esperada",
        "umbral_estandar",
        "umbral_orientado_a_sensibilidad",
        "calcular_umbral_youden",
        "modelo_1",
        "modelo_2"
    ],
    "valor": [
        NOMBRE_PROYECTO,
        VERSION_CUADERNO,
        FECHA_EJECUCION,
        str(RUTA_ARCHIVO_DATOS),
        VARIABLE_OBJETIVO,
        UMBRAL_ESTANDAR,
        UMBRAL_SENSIBLE,
        CALCULAR_UMBRAL_YOUDEN,
        MODELO_LOGISTICO,
        MODELO_RANDOM_FOREST
    ]
})

exportar_csv(
    tabla_parametros,
    "INI-tabla_control_de_parametros_del_estudio.csv",
    "ini",
    "Tabla de control con los parámetros generales definidos para la ejecución del cuaderno."
)

exportar_excel(
    tabla_parametros,
    "INI-tabla_control_de_parametros_del_estudio.xlsx",
    "ini",
    "Tabla de control con los parámetros generales definidos para la ejecución del cuaderno."
)

tabla_parametros

# ============================================================================
# 3.1 CARGA DEL ARCHIVO DE DATOS
# ============================================================================

USAR_KAGGLE_COMO_RESPALDO = True
ARCHIVO_DATOS = "Student Depression Dataset.csv"
RUTA_ARCHIVO_LOCAL = RUTAS["base_datos"] / ARCHIVO_DATOS

def cargar_dataset_desde_fuente():

    if RUTA_ARCHIVO_LOCAL.exists():
        df_cargado = pd.read_csv(RUTA_ARCHIVO_LOCAL)
```

```
        fuente = "archivo_local"
        return df_cargado, fuente

    if USAR_KAGGLE_COMO_RESPALDO:
        try:
            import kagglehub
            from kagglehub import KaggleDatasetAdapter

            df_cargado = kagglehub.dataset_load(
                KaggleDatasetAdapter.PANDAS,
                "hopesb/student-depression-dataset",
                ARCHIVO_DATOS
            )

            # Guardar copia local
            df_cargado.to_csv(RUTA_ARCHIVO_LOCAL, index=False, encoding="utf-8-sig")
            fuente = "kaggle"
            return df_cargado, fuente

        except Exception as e:
            raise RuntimeError(
                "No fue posible cargar el dataset desde Kaggle ni se encontró una copia local."
            ) from e

    raise FileNotFoundError(
        f"No se encontró el archivo local en: {RUTA_ARCHIVO_LOCAL}"
    )

df, fuente_datos = cargar_dataset_desde_fuente()
df_original = df.copy()

print("Archivo cargado correctamente.")
print("Fuente de datos utilizada:", fuente_datos)
print("Dimensiones del dataset:", df.shape)

# ============================================================================
# 3.3 TABLA UNIFICADA DE DIAGNÓSTICO DEL DATASET
# ============================================================================

diagnostico_dataset = df.dtypes.reset_index()
diagnostico_dataset.columns = ["variable", "tipo_dato"]

# Faltantes
faltantes = df.isnull().sum().reset_index()
faltantes.columns = ["variable", "cantidad_faltantes"]

# Merge
diagnostico_dataset = diagnostico_dataset.merge(
    faltantes,
    on="variable",
    how="left"
)

# Porcentaje
diagnostico_dataset["porcentaje_faltantes"] = (
    diagnostico_dataset["cantidad_faltantes"] / len(df)
)

diagnostico_dataset = diagnostico_dataset.sort_values(
```

```
    by="porcentaje_faltantes",
    ascending=False
)

diagnostico_dataset

"""## 3.4 Validación de la variable objetivo

Se verifica que la variable objetivo definida en los parámetros del estudio se encuentre
presente en el dataset cargado.

"""

if VARIABLE_OBJETIVO not in df.columns:
    raise ValueError(
        f"La variable objetivo '{VARIABLE_OBJETIVO}' no se encuentra en el dataset."
    )

print(f"Variable objetivo '{VARIABLE_OBJETIVO}' verificada correctamente.")

"""## 3.5 Detección de registros duplicados

Se verifica la existencia de registros duplicados en el dataset, ya que estos pueden afectar
los resultados del análisis y el desempeño de los modelos.
"""

duplicados = df.duplicated().sum()

print("Número de registros duplicados:", duplicados)

"""## 3.6 Resumen general del dataset

Se genera una tabla resumen con información general del dataset, incluyendo el número de
observaciones, el número de variables y el total de valores faltantes.
"""

resumen_dataset = pd.DataFrame({
    "metrica": [
        "numero_filas",
        "numero_columnas",
        "total_valores_faltantes"
    ],
    "valor": [
        df.shape[0],
        df.shape[1],
        df.isnull().sum().sum()
    ]
})

resumen_dataset

exportar_csv(
    diagnostico_dataset,
    "INI-diagnostico_general_del_dataset.csv",
    "ini",
    "Tabla consolidada con tipos de datos y valores faltantes por variable."
)

# Dimensiones del dataset
```

```
print("Número de observaciones:", df.shape[0])
print("Número de variables:", df.shape[1])

"""## 4.2 Estadísticas descriptivas de variables numéricas

Se calculan estadísticas descriptivas para las variables numéricas, con el fin de analizar su
distribución, tendencia central y dispersión.
"""

# Variables numéricas (excluyendo id y variable objetivo)
variables_numericas = df.select_dtypes(include=["int64", "float64"]).columns.tolist()

variables_numericas = [col for col in variables_numericas if col not in ["id",
VARIABLE_OBJETIVO]]

df[variables_numericas].describe()

exportar_csv(
    df[variables_numericas].describe().reset_index(),
    "EDA-estadisticas_variables_numericas.csv",
    "eda",
    "Estadísticas descriptivas de variables numéricas."
)

"""## 4.3 Estadísticas descriptivas de variables categóricas

Se analiza la distribución de las variables categóricas mediante tablas de frecuencia y
proporción.
"""

variables_categoricas = df.select_dtypes(include="object").columns.tolist()

for col in variables_categoricas:

    tabla = df[col].value_counts(dropna=False).reset_index()
    tabla.columns = [col, "frecuencia"]
    tabla["proporcion"] = tabla["frecuencia"] / len(df)

    display(tabla.head(10))

    exportar_csv(
        tabla,
        f"EDA-frecuencias_{limpiar_nombre_archivo(col)}.csv",
        "eda",
        f"Distribución de frecuencias de la variable {col}."
    )

"""## 4.4 Distribución de la variable objetivo

Se analiza la distribución de la variable objetivo con el fin de identificar la proporción de
casos positivos y negativos en el dataset.
"""

distribucion_objetivo = df[VARIABLE_OBJETIVO].value_counts().reset_index()
distribucion_objetivo.columns = ["clase", "frecuencia"]

distribucion_objetivo["proporcion"] = (
    distribucion_objetivo["frecuencia"] / len(df)
)
```

```
distribucion_objetivo


variables_revision = ["City", "Sleep Duration", "Dietary Habits", "Degree"]

for col in variables_revision:
    if col in df.columns:

        tabla = df[col].value_counts(dropna=False).reset_index()
        tabla.columns = [col, "frecuencia"]
        tabla["proporcion"] = tabla["frecuencia"] / len(df)

        display(tabla.head(15))

# ============================================================================
# 4.6 BOXPLOTS DE VARIABLES NUMÉRICAS
# ============================================================================

for col in variables_numericas:

    nombre_seguro = limpiar_nombre_archivo(col)

    plt.figure()
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot de {col}")

    exportar_grafica(
        f"EDA-boxplot_{nombre_seguro}.png",
        "eda",
        f"Boxplot de la variable {col}."
    )

    plt.show()
    plt.close()


# ============================================================================
# 5.1 CREACIÓN DE COPIA DE TRABAJO
# ============================================================================

df_clean = df.copy()

print("Copia de trabajo creada.")
print("Dimensiones iniciales:", df_clean.shape)

# ============================================================================
# 5.2 ELIMINACIÓN DE VARIABLES
# ============================================================================

variables_eliminar = [
    "id",
    "City",
    "Profession",
    "Work Pressure",
    "Job Satisfaction"
]

variables_existentes = [col for col in variables_eliminar if col in df_clean.columns]
```

```
df_clean = df_clean.drop(columns=variables_existentes)

print("Variables eliminadas:", variables_existentes)
print("Nueva dimensión:", df_clean.shape)

# ============================================================================
# 5.3 TRATAMIENTO DE "OTHERS"
# ============================================================================

variables_others = ["Sleep Duration", "Dietary Habits"]

registro_others = []

for col in variables_others:

    if col in df_clean.columns:

        moda = df_clean[col].mode()[0]
        cantidad = (df_clean[col] == "Others").sum()

        df_clean[col] = df_clean[col].replace("Others", moda)

        registro_others.append({
            "variable": col,
            "valor_reemplazado": "Others",
            "valor_asignado": moda,
            "cantidad_reemplazos": cantidad
        })

registro_others_df = pd.DataFrame(registro_others)

registro_others_df

# ============================================================================
# 5.4 IMPUTACIÓN DE FINANCIAL STRESS
# ============================================================================

mediana_fs = df_clean["Financial Stress"].median()

df_clean["Financial Stress"] = df_clean["Financial Stress"].fillna(mediana_fs)

print("Valores faltantes después de imputación:",
      df_clean["Financial Stress"].isnull().sum())

# ============================================================================
# 5.5 OUTLIERS EN CGPA
# ============================================================================

df_clean = df_clean[df_clean["CGPA"] > 0]
print("Filas eliminadas por CGPA=0:", (df["CGPA"] == 0).sum())
print("Dimensión después de eliminar CGPA = 0:", df_clean.shape)

"""## 5.6 Validación posterior a la limpieza

Se verifica la estructura final del dataset luego de aplicar las transformaciones de limpieza.
"""

df_clean.info()
```

```
df_clean.isnull().sum()

exportar_csv(
    df_clean,
    "LIM-base_de_datos_limpia.csv",
    "lim",
    "Base de datos final luego de limpieza y preparación."
)


# ============================================================================
# 6.1 BASE DE ANÁLISIS PARA SELECCIÓN DE VARIABLES
# ============================================================================

df_seleccion = df_clean.copy()

print("Dimensiones de la base para selección:", df_seleccion.shape)

"""## 6.2 Identificación de variables por tipo

Se identifican las variables numéricas y categóricas presentes en la base de datos limpia,
excluyendo la variable objetivo.
"""

# ============================================================================
# 6.2 IDENTIFICACIÓN DE VARIABLES NUMÉRICAS Y CATEGÓRICAS
# ============================================================================

variables_numericas_seleccion = df_seleccion.select_dtypes(include=["int64",
"float64"]).columns.tolist()
variables_numericas_seleccion = [
    col for col in variables_numericas_seleccion if col != VARIABLE_OBJETIVO
]

variables_categoricas_seleccion =
df_seleccion.select_dtypes(include=["object"]).columns.tolist()

print("Variables numéricas:")
print(variables_numericas_seleccion)

print("\nVariables categóricas:")
print(variables_categoricas_seleccion)

"""## 6.3 Relación entre variables categóricas y depresión

Se construyen tablas de contingencia normalizadas por fila para examinar la proporción de casos
con y sin depresión dentro de cada categoría.

"""

# ============================================================================
# GRÁFICAS VARIABLES CATEGÓRICAS VS DEPRESIÓN
# ============================================================================

for col in variables_categoricas_seleccion:

    tabla = pd.crosstab(
        df_seleccion[col],
```

```
        df_seleccion[VARIABLE_OBJETIVO],
        normalize="index"
    )

    tabla.plot(kind="bar", stacked=True, figsize=(8,5))

    plt.title(f"{col} vs Depresión")
    plt.ylabel("Proporción")
    plt.xlabel(col)

    nombre_archivo = f"EDA-grafica_{limpiar_nombre_archivo(col)}_vs_depresion.png"

    exportar_grafica(
        nombre_archivo,
        "eda",
        f"Gráfico de proporciones de depresión según la variable {col}."
    )

    plt.show()
    plt.close()

# ============================================================================
# ANÁLISIS COMPLEMENTARIO: PRUEBAS DE ASOCIACIÓN BIVARIADA
# ============================================================================
from scipy.stats import chi2_contingency, mannwhitneyu, shapiro

# Variables segun el conjunto principal definido en el modelado
num_vars_biv = ["Age", "Academic Pressure", "Study Satisfaction",
                "Work/Study Hours", "Financial Stress"]
cat_vars_biv = ["Gender", "Sleep Duration", "Dietary Habits",
                "Family History of Mental Illness"]

# ----- Categoricas: Chi-cuadrado de independencia -----
filas_cat = []
for col in cat_vars_biv:
    tabla = pd.crosstab(df_clean[col], df_clean[VARIABLE_OBJETIVO])
    chi2, p, gl, _ = chi2_contingency(tabla)
    filas_cat.append({
        "variable": col,
        "prueba": "Chi-cuadrado",
        "estadistico": round(chi2, 3),
        "gl": gl,
        "p_valor": "<0.001" if p < 0.001 else round(p, 4)
    })
tabla_pvalores_categoricas = pd.DataFrame(filas_cat)
print("Asociacion variables categoricas vs depresion (Chi-cuadrado):")
display(tabla_pvalores_categoricas)

# ----- Numericas: diagnostico de normalidad + Mann-Whitney U (no parametrica) -----
filas_num = []
for col in num_vars_biv:
    g0 = df_clean.loc[df_clean[VARIABLE_OBJETIVO] == 0, col]
    g1 = df_clean.loc[df_clean[VARIABLE_OBJETIVO] == 1, col]
    # Shapiro tiene tope ~5000; se evalua en submuestra SOLO como diagnostico de normalidad
    p_norm = shapiro(g0.sample(min(5000, len(g0)), random_state=RANDOM_STATE))[1]
    u, p = mannwhitneyu(g0, g1, alternative="two-sided")
    filas_num.append({
        "variable": col,
        "prueba": "Mann-Whitney U",
```

```
        "normal_aprox": bool(p_norm > 0.05),
        "p_valor": "<0.001" if p < 0.001 else round(p, 4)
    })
tabla_pvalores_numericas = pd.DataFrame(filas_num)
print("\nAsociacion variables numericas vs depresion (Mann-Whitney U):")
display(tabla_pvalores_numericas)

# Exportacion para trazabilidad
try:
    exportar_csv(tabla_pvalores_categoricas,
                 "BIV-pvalores_categoricas.csv", "biv",
                 "Valores p (Chi-cuadrado) de asociacion categorica con depresion.")
    exportar_csv(tabla_pvalores_numericas,
                 "BIV-pvalores_numericas.csv", "biv",
                 "Valores p (Mann-Whitney U) de asociacion numerica con depresion.")
except Exception as _e:
    print("Nota: exportar_csv no disponible, se omite exportacion.")


# ============================================================================
# BOXPLOTS VARIABLES NUMÉRICAS VS DEPRESIÓN
# ============================================================================

for col in variables_numericas_seleccion:

    plt.figure(figsize=(6,4))

    sns.boxplot(
        x=df_seleccion[VARIABLE_OBJETIVO],
        y=df_seleccion[col]
    )

    plt.title(f"{col} vs Depresión")
    plt.xlabel("Depresión")
    plt.ylabel(col)

    nombre_archivo = f"EDA-boxplot_{limpiar_nombre_archivo(col)}_vs_depresion.png"

    exportar_grafica(
        nombre_archivo,
        "eda",
        f"Boxplot de la variable {col} según depresión."
    )

    plt.show()
    plt.close()

"""## 6.5 Matriz de correlación entre variables numéricas

Se construye una matriz de correlación entre variables numéricas y la variable objetivo, con el
fin de examinar asociaciones lineales y detectar posibles relaciones entre predictores.

"""

# ============================================================================
# 6.5 MATRIZ DE CORRELACIÓN
# ============================================================================

plt.figure(figsize=(10, 8))
```

```
matriz_correlacion = df_seleccion[variables_numericas_seleccion + [VARIABLE_OBJETIVO]].corr()

sns.heatmap(
    matriz_correlacion,
    annot=True,
    cmap="coolwarm",
    fmt=".2f",
    square=True
)

plt.title("Matriz de correlación entre variables numéricas y depresión")

exportar_grafica(
    "EDA-matriz_de_correlacion_numericas_y_depresion.png",
    "eda",
    "Matriz de correlación entre variables numéricas y la variable objetivo."
)

plt.show()
plt.close()

# ============================================================================
# CORRELACIÓN NO PARAMÉTRICA DE SPEARMAN
# ============================================================================
import seaborn as sns
import matplotlib.pyplot as plt

cols_corr = ["Age", "Academic Pressure", "CGPA", "Study Satisfaction",
             "Work/Study Hours", "Financial Stress", VARIABLE_OBJETIVO]
cols_corr = [c for c in cols_corr if c in df_clean.columns]

corr_spearman = df_clean[cols_corr].corr(method="spearman")

plt.figure(figsize=(8, 6))
sns.heatmap(corr_spearman, annot=True, cmap="coolwarm", center=0, fmt=".2f")
plt.title("Correlacion de Spearman entre variables numericas y depresion")
plt.tight_layout()
plt.show()

print("Matriz de correlacion de Spearman:")
display(corr_spearman.round(2))

try:
    exportar_csv(corr_spearman.reset_index(),
                 "COR-spearman_numericas.csv", "cor",
                 "Matriz de correlacion de Spearman entre numericas y depresion.")
except Exception as _e:
    print("Nota: exportar_csv no disponible, se omite exportacion.")


## 6.6 Registro preliminar de decisión sobre variables

Se construye una tabla de apoyo para registrar la decisión preliminar sobre la inclusión,
exclusión o revisión adicional de las variables consideradas en el modelado.
"""

# ============================================================================
# 6.6 ACTUALIZACIÓN DE LA TABLA DE DECISIÓN DE VARIABLES
```

```
# ============================================================================

tabla_decision_variables = pd.DataFrame({
    "variable": variables_numericas_seleccion + variables_categoricas_seleccion,
    "tipo_variable": (
        ["numerica"] * len(variables_numericas_seleccion) +
        ["categorica"] * len(variables_categoricas_seleccion)
    ),
    "decision_preliminar": "mantener",
    "justificacion": ""
})

# ----------------------------------------------------------------------------
# AJUSTES ESPECÍFICOS SEGÚN EL ANÁLISIS EXPLORATORIO
# ----------------------------------------------------------------------------

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Degree",
    ["decision_preliminar", "justificacion"]
] = [
    "eliminar",
    "Se excluye por su alta cardinalidad, heterogeneidad conceptual y presencia de categorías
que tensionan la definición estricta de población universitaria, lo que reduce su utilidad
interpretativa en el modelo principal."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Have you ever had suicidal thoughts ?",
    ["decision_preliminar", "justificacion"]
] = [
    "excluir",
    "Se excluye del modelo principal por su alta proximidad conceptual con la variable
objetivo, con el fin de evitar redundancia predictiva y posible contaminación conceptual del
desenlace."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "CGPA",
    ["decision_preliminar", "justificacion"]
] = [
    "eliminar",
    "Se excluye del modelo principal porque su asociación exploratoria con la variable objetivo
fue limitada y, en la primera estimación logística, su inclusión se relacionó con colinealidad
elevada. Por ello, no se conserva en la especificación principal comparativa."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Gender",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Variable sociodemográfica de control incluida por relevancia sustantiva en el estudio."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Age",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
```

```
    "Presenta diferencias entre grupos y aporta información sociodemográfica relevante."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Academic Pressure",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Presenta asociación clara con la variable objetivo en el análisis exploratorio."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Study Satisfaction",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Se conserva por su relevancia conceptual y por diferencias observadas entre grupos."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Sleep Duration",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Presenta diferencias de proporción entre categorías respecto a la variable objetivo."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Dietary Habits",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Presenta diferencias claras entre categorías y capacidad explicativa relevante."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Work/Study Hours",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Presenta diferencias moderadas entre grupos y asociación útil para el modelado."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Financial Stress",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Presenta asociación importante con la variable objetivo en el análisis exploratorio."
]

tabla_decision_variables.loc[
    tabla_decision_variables["variable"] == "Family History of Mental Illness",
    ["decision_preliminar", "justificacion"]
] = [
    "mantener",
    "Se conserva por su relevancia sustantiva y por diferencias observadas entre categorías."
]
```

```
tabla_decision_variables

exportar_csv(
    tabla_decision_variables,
    "EDA-tabla_decision_de_variables_actualizada.csv",
    "eda",
    "Tabla actualizada de decisión preliminar sobre inclusión, exclusión o uso condicionado de
variables en el modelado."
)

## 7.1 Datasets del modelado
"""

# Modelo principal
variables_modelo_principal = [
    "Gender",
    "Age",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]

X_principal = df_clean[variables_modelo_principal].copy()
y = df_clean[VARIABLE_OBJETIVO].copy()

print("Modelo principal:", X_principal.shape)
print("Variable objetivo:", y.shape)

print("\nVariables del modelo principal:")
print(X_principal.columns.tolist())

# ============================================================================
# 7.2 IDENTIFICACIÓN DE VARIABLES POR TIPO
# ============================================================================

def separar_tipos_variables(X):
    numericas = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
    categoricas = X.select_dtypes(include=["object"]).columns.tolist()
    return numericas, categoricas

num_principal, cat_principal = separar_tipos_variables(X_principal)

print("Modelo principal:")
print("Numéricas:", num_principal)
print("Categóricas:", cat_principal)

tabla_tipos_modelado = pd.DataFrame({
    "dataset": ["principal"] * (len(num_principal) + len(cat_principal)),
    "variable": num_principal + cat_principal,
    "tipo": ["numerica"] * len(num_principal) + ["categorica"] * len(cat_principal)
})

display(tabla_tipos_modelado)

exportar_csv(
```

```
    tabla_tipos_modelado,
    "MLG-tabla_tipos_de_variables_modelado.csv",
    "mlg",
    "Clasificación de variables numéricas y categóricas para el conjunto principal de
modelado."
)

"""## 7.3 Pipeline de preprocesamiento

Se construye un pipeline que permite transformar las variables predictoras antes del modelado:

- Las variables numéricas son estandarizadas mediante StandardScaler.
- Las variables categóricas son codificadas mediante One-Hot Encoding, eliminando una categoría
base para evitar colinealidad perfecta.

"""

# ============================================================================
# 7.3 PIPELINE DE PREPROCESAMIENTO
# ============================================================================

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler

def crear_pipeline(numericas, categoricas):

    transformador_numerico = Pipeline(steps=[
        ("scaler", StandardScaler())
    ])

    transformador_categorico = Pipeline(steps=[
        ("onehot", OneHotEncoder(drop="first", handle_unknown="ignore"))
    ])

    preprocesador = ColumnTransformer(
        transformers=[
            ("num", transformador_numerico, numericas),
            ("cat", transformador_categorico, categoricas)
        ]
    )

    return preprocesador

# Pipelines
pipeline_principal = crear_pipeline(num_principal, cat_principal)

print("Pipelines creados correctamente.")

"""## 7.4 Validación del Pipeline"""

# ============================================================================
# 7.4 VALIDACIÓN DEL PIPELINE
# ============================================================================

X_principal_transf = pipeline_principal.fit_transform(X_principal)

print("Forma modelo principal:", X_principal_transf.shape)
```

```
"""### 7.4.1 Nota de trazabilidad sobre las dimensiones de modelado

"""

# ============================================================================
# Verificación final de trazabilidad antes del modelado
# ============================================================================

print("X_principal crudo:", X_principal.shape)

X_principal_transf = pipeline_principal.fit_transform(X_principal)

print("X_principal transformado:", X_principal_transf.shape)

"""## 7.5 Esquema de validación del modelado

"""

# ============================================================================
# 7.5 NOMBRES DE VARIABLES TRANSFORMADAS
# ============================================================================

feature_names_principal = pipeline_principal.get_feature_names_out()

print("Variables transformadas (principal):")
print(feature_names_principal[:20])

print("\nTotal variables principal:", len(feature_names_principal))

estructura_final = pd.DataFrame({
    "variable_transformada_principal": feature_names_principal
})

exportar_csv(
    estructura_final,
    "MLG-variables_transformadas_modelo_principal.csv",
    "mlg",
    "Listado de variables generadas tras el pipeline de preprocesamiento (modelo principal)."
)

print("Categoricas principal:", cat_principal)
print("Numericas principal:", num_principal)
print(len(feature_names_principal))
print(feature_names_principal)

"""## 7.6 Métricas de evaluación y criterio de lectura

"""

print("X_principal crudo:", X_principal.shape)

print("\nColumnas X_principal:")
print(X_principal.columns.tolist())

X_principal_transf = pipeline_principal.fit_transform(X_principal)

print("X_principal transformado:", X_principal_transf.shape)
```

Código B 2. Regresión logística explicativa

Este fragmento presenta la construcción de la matriz de diseño, la codificación de variables categóricas, la definición de categorías de referencia, el diagnóstico de colinealidad mediante VIF y el ajuste del modelo logístico explicativo con statsmodels.Logit. También incluye la comparación de especificaciones con y sin la variable Age. El modelo se utiliza para interpretar asociaciones ajustadas mediante coeficientes, valores p, Odds Ratios e intervalos de confianza.

```
# ============================================================================
# MODELADO - REGRESION LOGISTICA EXPLICATIVA
# ============================================================================

# ============================================================================
# 8.1.0 FUNCIONES AUXILIARES DE FORMATO
# ============================================================================

import numpy as np
import pandas as pd

def formatear_pvalor(p):
    if pd.isna(p):
        return ""
    if p < 0.001:
        return "<0.001"
    return f"{p:.4f}"

def redondear_tabla(df, columnas_4=None, columnas_3=None):
    df_out = df.copy()

    if columnas_4 is not None:
        for col in columnas_4:
            if col in df_out.columns:
                df_out[col] = pd.to_numeric(df_out[col], errors="coerce").round(4)

    if columnas_3 is not None:
        for col in columnas_3:
            if col in df_out.columns:
                df_out[col] = pd.to_numeric(df_out[col], errors="coerce").round(3)

    return df_out

# ============================================================================
# 8.1.1 VERIFICACIÓN DE INSUMOS DEL MODELO EXPLICATIVO
# ============================================================================

import statsmodels.api as sm
from IPython.display import display

print("X_principal crudo:", X_principal.shape)
print("y:", y.shape)

print("\nVariables del modelo principal:")
print(X_principal.columns.tolist())
```

```
print("\nVariables numéricas:")
print(num_principal)

print("\nVariables categóricas:")
print(cat_principal)

# Copias de trabajo
X_logit_exp_base = X_principal.copy()
y_logit_exp = y.copy()

# ============================================================================
# 8.1.1 CONSTRUCCIÓN DE MATRIZ DE DISEÑO
# ============================================================================

import pandas as pd
import statsmodels.api as sm

df_logit_exp = X_logit_exp_base.copy()

referencias_categoricas = {
    "Gender": ["Female", "Male"],
    "Sleep Duration": ["5-6 hours", "7-8 hours", "Less than 5 hours", "More than 8 hours"],
    "Dietary Habits": ["Healthy", "Moderate", "Unhealthy"],
    "Family History of Mental Illness": ["No", "Yes"],
}

for var, orden_esperado in referencias_categoricas.items():
    valores_presentes = df_logit_exp[var].astype(str).unique().tolist()
    categorias_finales = [c for c in orden_esperado if c in valores_presentes]

    if len(categorias_finales) == 0:
        categorias_finales = sorted(valores_presentes)

    df_logit_exp[var] = pd.Categorical(
        df_logit_exp[var].astype(str),
        categories=categorias_finales,
        ordered=True
    )

X_logit_exp = pd.get_dummies(
    df_logit_exp,
    columns=cat_principal,
    drop_first=True,
    dtype=float
)

y_logit_exp = pd.to_numeric(y_logit_exp, errors="raise").astype(int)

X_logit_exp = sm.add_constant(X_logit_exp, has_constant="add")

tabla_variables_logit_exp = pd.DataFrame({"variable": X_logit_exp.columns})

print("Forma final de X_logit_exp:", X_logit_exp.shape)
print("Forma final de y_logit_exp:", y_logit_exp.shape)
display(tabla_variables_logit_exp)

exportar_csv(
    tabla_variables_logit_exp,
    "MLG-variables_finales_modelo_logistico_explicativo.csv",
```

```
    "mlg",
    "Listado final de variables incluidas en la matriz de diseño del modelo logístico
explicativo."
)

# ============================================================================
# 8.1.2 DIAGNÓSTICO DE MULTICOLINEALIDAD (VIF)
# ============================================================================

from statsmodels.stats.outliers_influence import variance_inflation_factor

X_vif = X_logit_exp.drop(columns="const").copy()

tabla_vif_logit_exp = pd.DataFrame({
    "variable": X_vif.columns,
    "VIF": [variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]
}).sort_values("VIF", ascending=False).reset_index(drop=True)

tabla_vif_logit_exp_export = tabla_vif_logit_exp.copy()
tabla_vif_logit_exp_vista = redondear_tabla(tabla_vif_logit_exp, columnas_3=["VIF"])

display(tabla_vif_logit_exp_vista)

exportar_csv(
    tabla_vif_logit_exp_export,
    "MLG-vif_modelo_logistico_explicativo.csv",
    "mlg",
    "Diagnóstico de multicolinealidad mediante VIF para el modelo logístico explicativo."
)

"""### 8.1.3 Estimación del modelo

Una vez construida la matriz de diseño y revisada la colinealidad entre predictores, se procede
al ajuste del modelo logístico explicativo.

"""

# ============================================================================
# 8.1.4 AJUSTE DEL MODELO LOGÍSTICO EXPLICATIVO
# ============================================================================

import statsmodels.api as sm

modelo_logit_exp = sm.Logit(y_logit_exp, X_logit_exp)

try:
    resultado_logit_exp = modelo_logit_exp.fit(
        method="lbfgs",
        maxiter=1000,
        disp=False
    )
except Exception as e:
    print("Primer intento falló:", repr(e))
    print("Se intentará nuevamente con método 'newton'.")
    resultado_logit_exp = modelo_logit_exp.fit(
        method="newton",
        maxiter=300,
        disp=False
    )
```

```
print("Convergencia del modelo:", resultado_logit_exp.mle_retvals.get("converged", "no
disponible"))
print(resultado_logit_exp.summary2())

tabla_summary_modelo = resultado_logit_exp.summary2().tables[0].reset_index()
tabla_summary_modelo.columns = [str(c).strip().lower().replace(" ", "_") for c in
tabla_summary_modelo.columns]

tabla_summary_coef = resultado_logit_exp.summary2().tables[1].reset_index()
tabla_summary_coef = tabla_summary_coef.rename(columns={"index": "variable"})

exportar_csv(
    tabla_summary_modelo,
    "MLG-resumen_general_modelo_logistico_explicativo.csv",
    "mlg",
    "Resumen general del ajuste del modelo logístico explicativo."
)

exportar_csv(
    tabla_summary_coef,
    "MLG-resumen_coeficientes_modelo_logistico_explicativo.csv",
    "mlg",
    "Resumen técnico de coeficientes del modelo logístico explicativo generado por
statsmodels."
)

"""### 8.1.4 Especificación inicial A (diagnóstico)

"""

# ============================================================================
# 8.1.4 TABLA DE COEFICIENTES Y ODDS RATIOS
# ============================================================================

ic = resultado_logit_exp.conf_int()

tabla_logit_exp = pd.DataFrame({
    "variable": resultado_logit_exp.params.index,
    "coef": resultado_logit_exp.params.values,
    "error_std": resultado_logit_exp.bse.values,
    "z": resultado_logit_exp.tvalues.values,
    "p_value": resultado_logit_exp.pvalues.values,
    "ic95_coef_inf": ic[0].values,
    "ic95_coef_sup": ic[1].values
})

tabla_logit_exp["odds_ratio"] = np.exp(tabla_logit_exp["coef"])
tabla_logit_exp["ic95_or_inf"] = np.exp(tabla_logit_exp["ic95_coef_inf"])
tabla_logit_exp["ic95_or_sup"] = np.exp(tabla_logit_exp["ic95_coef_sup"])
tabla_logit_exp["significativo_5pct"] = tabla_logit_exp["p_value"] < 0.05

exportar_csv(
    tabla_logit_exp,
    "MLG-coeficientes_modelo_logistico_explicativo.csv",
    "mlg",
    "Coeficientes, errores estándar, estadísticos z, valores p e intervalos de confianza del
modelo logístico explicativo."
)
```

```
tabla_logit_exp_vista = redondear_tabla(
    tabla_logit_exp,
    columnas_4=[
        "coef", "error_std", "z",
        "ic95_coef_inf", "ic95_coef_sup",
        "odds_ratio", "ic95_or_inf", "ic95_or_sup"
    ]
)
tabla_logit_exp_vista["p_value"] = tabla_logit_exp_vista["p_value"].apply(formatear_pvalor)

display(tabla_logit_exp_vista)

"""### 8.1.5 Lectura diagnóstica inicial"""

# ============================================================================
# 8.1.5 TABLA COMPACTA DE INTERPRETACIÓN
# ============================================================================

tabla_logit_exp_compacta = (
    tabla_logit_exp.loc[tabla_logit_exp["variable"] != "const", [
        "variable", "coef", "odds_ratio", "p_value",
        "ic95_or_inf", "ic95_or_sup", "significativo_5pct"
    ]]
    .sort_values("odds_ratio", ascending=False)
    .reset_index(drop=True)
)

exportar_csv(
    tabla_logit_exp_compacta,
    "MLG-tabla_compacta_modelo_logistico_explicativo.csv",
    "mlg",
    "Tabla compacta de interpretación del modelo logístico explicativo."
)

tabla_logit_exp_compacta_vista = redondear_tabla(
    tabla_logit_exp_compacta,
    columnas_4=["coef", "odds_ratio", "ic95_or_inf", "ic95_or_sup"]
)
tabla_logit_exp_compacta_vista["p_value"] =
tabla_logit_exp_compacta_vista["p_value"].apply(formatear_pvalor)

display(tabla_logit_exp_compacta_vista)


# ============================================================================
# 8.1.6 INDICADORES GLOBALES DE AJUSTE
# ============================================================================

ajuste_logit_exp = pd.DataFrame([{
    "n_obs": int(resultado_logit_exp.nobs),
    "llf": resultado_logit_exp.llf,
    "llnull": resultado_logit_exp.llnull,
    "pseudo_r2_mcfadden": resultado_logit_exp.prsquared,
    "aic": resultado_logit_exp.aic,
    "bic": resultado_logit_exp.bic
}])

exportar_csv(
```

```
    ajuste_logit_exp,
    "MLG-indicadores_globales_ajuste_modelo_logistico_explicativo.csv",
    "mlg",
    "Indicadores globales de ajuste del modelo logístico explicativo."
)

ajuste_logit_exp_vista = redondear_tabla(
    ajuste_logit_exp,
    columnas_4=["llf", "llnull", "pseudo_r2_mcfadden", "aic", "bic"]
)

display(ajuste_logit_exp_vista)


# ============================================================================
# 8.1.7.1 DIAGNÓSTICO REFORZADO DE COLINEALIDAD - CORRELACIÓN NUMÉRICA
# ============================================================================

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

tabla_corr_numericas_81 = X_principal[num_principal].corr()

display(tabla_corr_numericas_81.round(3))

exportar_csv(
    tabla_corr_numericas_81,
    "MLG-81_correlacion_variables_numericas.csv",
    "mlg",
    "Matriz de correlación entre variables numéricas del modelo principal para diagnóstico de
colinealidad."
)

plt.figure(figsize=(8, 6))
sns.heatmap(tabla_corr_numericas_81, annot=True, cmap="coolwarm", center=0, fmt=".2f")
plt.title("Correlación entre variables numéricas - Modelo principal")
plt.tight_layout()
plt.show()

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

referencias_categoricas = {
    "Gender": ["Female", "Male"],
    "Sleep Duration": ["5-6 hours", "7-8 hours", "Less than 5 hours", "More than 8 hours"],
    "Dietary Habits": ["Healthy", "Moderate", "Unhealthy"],
    "Family History of Mental Illness": ["No", "Yes"],
}

def construir_matriz_logit(df_base, columnas_categoricas):
    df_tmp = df_base.copy()

    for var, orden_esperado in referencias_categoricas.items():
        if var in df_tmp.columns:
            valores_presentes = df_tmp[var].astype(str).unique().tolist()
            categorias_finales = [c for c in orden_esperado if c in valores_presentes]
            if len(categorias_finales) == 0:
```

```
                categorias_finales = sorted(valores_presentes)

            df_tmp[var] = pd.Categorical(
                df_tmp[var].astype(str),
                categories=categorias_finales,
                ordered=True
            )

    X = pd.get_dummies(
        df_tmp,
        columns=[c for c in columnas_categoricas if c in df_tmp.columns],
        drop_first=True,
        dtype=float
    )

    X = sm.add_constant(X, has_constant="add")
    return X

def calcular_vif_y_condicion(X):
    X_sin_const = X.drop(columns="const", errors="ignore").copy()

    tabla_vif = pd.DataFrame({
        "variable": X_sin_const.columns,
        "VIF": [variance_inflation_factor(X_sin_const.values, i) for i in
range(X_sin_const.shape[1])]
    }).sort_values("VIF", ascending=False).reset_index(drop=True)

    # número de condición global
    condicion = np.linalg.cond(X_sin_const.values)

    return tabla_vif, condicion

# ============================================================================
# 8.1.7.2 MODELO A - DIAGNÓSTICO DE LA ESPECIFICACIÓN ACTUAL
# ============================================================================

variables_logit_A = [
    "Gender",
    "Age",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]

df_logit_A = df_clean[variables_logit_A].copy()
X_logit_A = construir_matriz_logit(df_logit_A, cat_principal)

tabla_vif_A, condicion_A = calcular_vif_y_condicion(X_logit_A)

print("Número de condición - Modelo A:", round(condicion_A, 2))
display(tabla_vif_A.round(3))

exportar_csv(
    tabla_vif_A,
    "MLG-81_vif_modelo_A_actual.csv",
```

```
    "mlg",
    "Diagnóstico VIF de la especificación explicativa actual (Modelo A)."
)

tabla_condicion_A = pd.DataFrame([{
    "modelo": "A_actual",
    "numero_condicion": condicion_A
}])

display(tabla_condicion_A.round(2))

exportar_csv(
    tabla_condicion_A,
    "MLG-81_numero_condicion_modelo_A_actual.csv",
    "mlg",
    "Número de condición de la especificación explicativa actual (Modelo A)."
)

"""#### 8.1.7.3 Diagnostico Modelo B (Sin Age)"""

# ============================================================================
# 8.1.7.3 MODELO B - DIAGNÓSTICO SIN AGE
# ============================================================================

variables_logit_B = [
    "Gender",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]

cat_logit_B = ["Gender", "Sleep Duration", "Dietary Habits", "Family History of Mental
Illness"]

df_logit_B = df_clean[variables_logit_B].copy()
X_logit_B = construir_matriz_logit(df_logit_B, cat_logit_B)

tabla_vif_B, condicion_B = calcular_vif_y_condicion(X_logit_B)

print("Número de condición - Modelo B:", round(condicion_B, 2))
display(tabla_vif_B.round(3))

exportar_csv(
    tabla_vif_B,
    "MLG-81_vif_modelo_B_sin_age.csv",
    "mlg",
    "Diagnóstico VIF de la especificación explicativa sin Age (Modelo B)."
)

tabla_condicion_B = pd.DataFrame([{
    "modelo": "B_sin_age",
    "numero_condicion": condicion_B
}])

display(tabla_condicion_B.round(2))
```

```
exportar_csv(
    tabla_condicion_B,
    "MLG-81_numero_condicion_modelo_B_sin_age.csv",
    "mlg",
    "Número de condición de la especificación explicativa sin Age (Modelo B)."
)

"""#### 8.1.7.4 Comparación de Colinealidad (A y B)"""

# ============================================================================
# 8.1.7.4 COMPARACIÓN DE COLINEALIDAD ENTRE MODELOS A Y B
# ============================================================================

tabla_comp_colinealidad_AB = pd.DataFrame([
    {
        "modelo": "A_actual",
        "vif_maximo": tabla_vif_A["VIF"].max(),
        "n_vif_mayor_5": int((tabla_vif_A["VIF"] > 5).sum()),
        "numero_condicion": condicion_A
    },
    {
        "modelo": "B_sin_age",
        "vif_maximo": tabla_vif_B["VIF"].max(),
        "n_vif_mayor_5": int((tabla_vif_B["VIF"] > 5).sum()),
        "numero_condicion": condicion_B
    }
])

tabla_comp_colinealidad_AB_vista = tabla_comp_colinealidad_AB.round(3)
display(tabla_comp_colinealidad_AB_vista)

exportar_csv(
    tabla_comp_colinealidad_AB,
    "MLG-81_comparacion_colinealidad_modelos_A_B.csv",
    "mlg",
    "Comparación del diagnóstico de colinealidad entre la especificación actual y la
especificación sin Age."
)

# ============================================================================
# 8.1.8 FUNCIONES AUXILIARES PARA AJUSTAR ESPECIFICACIONES EXPLICATIVAS
# ============================================================================

import numpy as np
import pandas as pd
import statsmodels.api as sm

def ajustar_modelo_logit_explicativo(nombre_modelo, variables_modelo, columnas_categoricas):
    df_tmp = df_clean[variables_modelo].copy()
    X_tmp = construir_matriz_logit(df_tmp, columnas_categoricas)
    y_tmp = y.copy().astype(int)

    tabla_vif_tmp, condicion_tmp = calcular_vif_y_condicion(X_tmp)

    modelo_tmp = sm.Logit(y_tmp, X_tmp)

    try:
        resultado_tmp = modelo_tmp.fit(method="lbfgs", maxiter=1000, disp=False)
```

```
    except Exception as e:
        print(f"{nombre_modelo}: fallo con lbfgs -> {repr(e)}")
        resultado_tmp = modelo_tmp.fit(method="newton", maxiter=300, disp=False)

    ic_tmp = resultado_tmp.conf_int()

    tabla_coef_tmp = pd.DataFrame({
        "modelo": nombre_modelo,
        "variable": resultado_tmp.params.index,
        "coef": resultado_tmp.params.values,
        "error_std": resultado_tmp.bse.values,
        "z": resultado_tmp.tvalues.values,
        "p_value": resultado_tmp.pvalues.values,
        "ic95_coef_inf": ic_tmp[0].values,
        "ic95_coef_sup": ic_tmp[1].values
    })

    tabla_coef_tmp["odds_ratio"] = np.exp(tabla_coef_tmp["coef"])
    tabla_coef_tmp["ic95_or_inf"] = np.exp(tabla_coef_tmp["ic95_coef_inf"])
    tabla_coef_tmp["ic95_or_sup"] = np.exp(tabla_coef_tmp["ic95_coef_sup"])
    tabla_coef_tmp["significativo_5pct"] = tabla_coef_tmp["p_value"] < 0.05

    resumen_tmp = {
        "modelo": nombre_modelo,
        "n_predictores": X_tmp.shape[1] - 1,
        "vif_maximo": float(tabla_vif_tmp["VIF"].max()),
        "n_vif_mayor_5": int((tabla_vif_tmp["VIF"] > 5).sum()),
        "numero_condicion": float(condicion_tmp),
        "llf": float(resultado_tmp.llf),
        "llnull": float(resultado_tmp.llnull),
        "pseudo_r2_mcfadden": float(resultado_tmp.prsquared),
        "aic": float(resultado_tmp.aic),
        "bic": float(resultado_tmp.bic),
        "converged": resultado_tmp.mle_retvals.get("converged", np.nan)
    }

    return {
        "X": X_tmp,
        "resultado": resultado_tmp,
        "tabla_vif": tabla_vif_tmp,
        "tabla_coef": tabla_coef_tmp,
        "resumen": resumen_tmp
    }

# ============================================================================
# 8.1.8 AJUSTE DE LAS ESPECIFICACIONES B, C1 Y C2
# ============================================================================

# Modelo B: sin Age
variables_logit_B = [
    "Gender",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]
```

```
cat_logit_B = ["Gender", "Sleep Duration", "Dietary Habits", "Family History of Mental
Illness"]

# Modelo C1: sin Age y sin Academic Pressure
variables_logit_C1 = [
    "Gender",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]
cat_logit_C1 = ["Gender", "Sleep Duration", "Dietary Habits", "Family History of Mental
Illness"]

# Modelo C2: sin Age y sin Financial Stress
variables_logit_C2 = [
    "Gender",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Family History of Mental Illness"
]
cat_logit_C2 = ["Gender", "Sleep Duration", "Dietary Habits", "Family History of Mental
Illness"]

ajuste_B = ajustar_modelo_logit_explicativo("B_sin_age", variables_logit_B, cat_logit_B)
ajuste_C1 = ajustar_modelo_logit_explicativo("C1_sin_age_sin_academic_pressure",
variables_logit_C1, cat_logit_C1)
ajuste_C2 = ajustar_modelo_logit_explicativo("C2_sin_age_sin_financial_stress",
variables_logit_C2, cat_logit_C2)

print("Modelos ajustados correctamente.")

# ============================================================================
# 8.1.8 TABLA COMPARATIVA DE ESPECIFICACIONES EXPLICATIVAS
# ============================================================================

tabla_comparativa_especificaciones_81 = pd.DataFrame([
    ajuste_B["resumen"],
    ajuste_C1["resumen"],
    ajuste_C2["resumen"]
])

tabla_comparativa_especificaciones_81_vista = tabla_comparativa_especificaciones_81.copy()
for col in ["vif_maximo", "numero_condicion", "llf", "llnull", "pseudo_r2_mcfadden", "aic",
"bic"]:
    tabla_comparativa_especificaciones_81_vista[col] =
tabla_comparativa_especificaciones_81_vista[col].round(4)

display(tabla_comparativa_especificaciones_81_vista)

exportar_csv(
    tabla_comparativa_especificaciones_81,
    "MLG-81_comparacion_especificaciones_explicativas.csv",
    "mlg",
```

```
    "Comparación entre las especificaciones explicativas B, C1 y C2 en términos de colinealidad
y ajuste."
)

# ============================================================================
# 8.1.8 VIF DE CADA ESPECIFICACIÓN
# ============================================================================

tabla_vif_B_vista = ajuste_B["tabla_vif"].round(3)
tabla_vif_C1_vista = ajuste_C1["tabla_vif"].round(3)
tabla_vif_C2_vista = ajuste_C2["tabla_vif"].round(3)

print("VIF - Modelo B")
display(tabla_vif_B_vista)

print("VIF - Modelo C1")
display(tabla_vif_C1_vista)

print("VIF - Modelo C2")
display(tabla_vif_C2_vista)

exportar_csv(
    ajuste_B["tabla_vif"],
    "MLG-81_vif_modelo_B_finalista.csv",
    "mlg",
    "VIF del modelo explicativo B."
)

exportar_csv(
    ajuste_C1["tabla_vif"],
    "MLG-81_vif_modelo_C1_finalista.csv",
    "mlg",
    "VIF del modelo explicativo C1."
)

exportar_csv(
    ajuste_C2["tabla_vif"],
    "MLG-81_vif_modelo_C2_finalista.csv",
    "mlg",
    "VIF del modelo explicativo C2."
)

# ============================================================================
# 8.1.8 TABLAS COMPACTAS DE COEFICIENTES POR ESPECIFICACIÓN
# ============================================================================

def preparar_tabla_compacta_coef(ajuste_dict):
    tabla = ajuste_dict["tabla_coef"].copy()
    tabla = tabla.loc[tabla["variable"] != "const", [
        "modelo", "variable", "coef", "odds_ratio", "p_value",
        "ic95_or_inf", "ic95_or_sup", "significativo_5pct"
    ]].copy()
    return tabla.sort_values("odds_ratio", ascending=False).reset_index(drop=True)

tabla_coef_B_compacta = preparar_tabla_compacta_coef(ajuste_B)
tabla_coef_C1_compacta = preparar_tabla_compacta_coef(ajuste_C1)
tabla_coef_C2_compacta = preparar_tabla_compacta_coef(ajuste_C2)

print("Coeficientes compactos - Modelo B")
```

```
display(redondear_tabla(tabla_coef_B_compacta, columnas_4=["coef", "odds_ratio", "ic95_or_inf",
"ic95_or_sup"]))

print("Coeficientes compactos - Modelo C1")
display(redondear_tabla(tabla_coef_C1_compacta, columnas_4=["coef", "odds_ratio",
"ic95_or_inf", "ic95_or_sup"]))

print("Coeficientes compactos - Modelo C2")
display(redondear_tabla(tabla_coef_C2_compacta, columnas_4=["coef", "odds_ratio",
"ic95_or_inf", "ic95_or_sup"]))

# ============================================================================
# 8.1.9.1 TABLA FINAL DEL MODELO EXPLICATIVO B_SIN_AGE
# ============================================================================

tabla_logit_B_final = ajuste_B["tabla_coef"].copy()

tabla_logit_B_final_vista = tabla_logit_B_final.copy()
for col in ["coef", "error_std", "z", "ic95_coef_inf", "ic95_coef_sup", "odds_ratio",
"ic95_or_inf", "ic95_or_sup"]:
    tabla_logit_B_final_vista[col] = tabla_logit_B_final_vista[col].round(4)

tabla_logit_B_final_vista["p_value"] =
tabla_logit_B_final_vista["p_value"].apply(formatear_pvalor)

display(tabla_logit_B_final_vista)

exportar_csv(
    tabla_logit_B_final,
    "MLG-81_coeficientes_modelo_logistico_explicativo_final_B.csv",
    "mlg",
    "Coeficientes, errores estándar, valores p, intervalos de confianza y odds ratios del
modelo explicativo final B_sin_age."
)
```

Código B 3. Regresión logística Ridge

El Código B3 presenta el fragmento correspondiente al modelo de regresión logística regularizada tipo Ridge. Incluye la partición entrenamiento/prueba, la construcción del pipeline, la validación cruzada para la selección del hiperparámetro C, el ajuste del modelo, la estimación de probabilidades, el cálculo de métricas globales, la curva ROC, la calibración y la evaluación de umbrales de clasificación.

```
# ============================================================================
# 8.2.1 DEFINICIÓN DEL ESQUEMA DE ENTRENAMIENTO Y PRUEBA
# ============================================================================

from sklearn.model_selection import train_test_split

try:
    random_state_modelado = RANDOM_STATE
except NameError:
    try:
        random_state_modelado = SEED
    except NameError:
        random_state_modelado = 42

test_size_modelado = 0.20
```

```
X_train_principal, X_test_principal, y_train_principal, y_test_principal = train_test_split(
    X_principal,
    y,
    test_size=test_size_modelado,
    stratify=y,
    random_state=random_state_modelado
)

tabla_split_principal = pd.DataFrame({
    "conjunto": ["X_train_principal", "X_test_principal", "y_train_principal",
"y_test_principal"],
    "filas": [
        X_train_principal.shape[0],
        X_test_principal.shape[0],
        y_train_principal.shape[0],
        y_test_principal.shape[0]
    ],
    "columnas": [
        X_train_principal.shape[1],
        X_test_principal.shape[1],
        1,
        1
    ]
})

display(tabla_split_principal)

exportar_csv(
    tabla_split_principal,
    "MLG-ridge_split_entrenamiento_prueba.csv",
    "mlg",
    "Dimensiones de los conjuntos de entrenamiento y prueba para la regresión logística Ridge."
)

# ============================================================================
# 8.2.2 CONSTRUCCIÓN DEL PIPELINE RIDGE Y BÚSQUEDA DE HIPERPARÁMETROS
# ============================================================================

from sklearn.base import clone
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline

try:
    n_splits_cv = N_SPLITS_CV
except NameError:
    n_splits_cv = 5

cv_ridge = StratifiedKFold(
    n_splits=n_splits_cv,
    shuffle=True,
    random_state=random_state_modelado
)

pipe_logit_ridge = Pipeline(steps=[
    ("preprocess", clone(pipeline_principal)),
    ("model", LogisticRegression(
        penalty="l2",
```

```
        solver="lbfgs",
        max_iter=5000,
        random_state=random_state_modelado
    ))
])

param_grid_ridge = {
    "model__C": np.logspace(-3, 3, 13)
}

grid_ridge = GridSearchCV(
    estimator=pipe_logit_ridge,
    param_grid=param_grid_ridge,
    scoring="roc_auc",
    cv=cv_ridge,
    n_jobs=-1,
    refit=True,
    return_train_score=True
)

grid_ridge.fit(X_train_principal, y_train_principal)

print("Mejor parámetro C:", grid_ridge.best_params_["model__C"])
print("Mejor AUC promedio en CV:", round(grid_ridge.best_score_, 4))


# ============================================================================
# 8.2.3 RESULTADOS DE LA VALIDACIÓN CRUZADA
# ============================================================================

tabla_cv_ridge = pd.DataFrame(grid_ridge.cv_results_)[[
    "param_model__C",
    "mean_train_score",
    "std_train_score",
    "mean_test_score",
    "std_test_score",
    "rank_test_score"
]].sort_values("rank_test_score").reset_index(drop=True)

tabla_cv_ridge_export = tabla_cv_ridge.copy()

tabla_cv_ridge_vista = tabla_cv_ridge.copy()
tabla_cv_ridge_vista["mean_train_score"] =
pd.to_numeric(tabla_cv_ridge_vista["mean_train_score"]).round(4)
tabla_cv_ridge_vista["std_train_score"] =
pd.to_numeric(tabla_cv_ridge_vista["std_train_score"]).round(4)
tabla_cv_ridge_vista["mean_test_score"] =
pd.to_numeric(tabla_cv_ridge_vista["mean_test_score"]).round(4)
tabla_cv_ridge_vista["std_test_score"] =
pd.to_numeric(tabla_cv_ridge_vista["std_test_score"]).round(4)

display(tabla_cv_ridge_vista)

exportar_csv(
    tabla_cv_ridge_export,
    "MLG-ridge_resultados_validacion_cruzada.csv",
    "mlg",
    "Resultados de validación cruzada para la selección del hiperparámetro C en la regresión
logística Ridge."
```

```
)

tabla_best_ridge = pd.DataFrame([{
    "mejor_C": grid_ridge.best_params_["model__C"],
    "mejor_auc_cv": grid_ridge.best_score_
}])

display(tabla_best_ridge.round(4))

exportar_csv(
    tabla_best_ridge,
    "MLG-ridge_mejor_hiperparametro.csv",
    "mlg",
    "Mejor hiperparámetro y desempeño promedio de validación cruzada del modelo logístico
Ridge."
)

# ============================================================================
# 8.2.4 COEFICIENTES REGULARIZADOS DEL MODELO SELECCIONADO
# ============================================================================

mejor_modelo_ridge = grid_ridge.best_estimator_

feature_names_ridge = mejor_modelo_ridge.named_steps["preprocess"].get_feature_names_out()
coef_ridge = mejor_modelo_ridge.named_steps["model"].coef_[0]

tabla_coef_ridge = pd.DataFrame({
    "variable_transformada": feature_names_ridge,
    "coef_ridge": coef_ridge,
    "abs_coef_ridge": np.abs(coef_ridge)
}).sort_values("abs_coef_ridge", ascending=False).reset_index(drop=True)

tabla_coef_ridge_vista = tabla_coef_ridge.copy()
tabla_coef_ridge_vista["coef_ridge"] = tabla_coef_ridge_vista["coef_ridge"].round(4)
tabla_coef_ridge_vista["abs_coef_ridge"] = tabla_coef_ridge_vista["abs_coef_ridge"].round(4)

display(tabla_coef_ridge_vista)

exportar_csv(
    tabla_coef_ridge,
    "MLG-ridge_coeficientes_regularizados.csv",
    "mlg",
    "Coeficientes regularizados del modelo logístico Ridge seleccionado."
)

# ============================================================================
# 8.2.5 GENERACIÓN DE PROBABILIDADES PREDICHAS
# ============================================================================

proba_train_ridge = mejor_modelo_ridge.predict_proba(X_train_principal)[:, 1]
proba_test_ridge = mejor_modelo_ridge.predict_proba(X_test_principal)[:, 1]

pred_test_ridge_050 = (proba_test_ridge >= 0.50).astype(int)
pred_test_ridge_020 = (proba_test_ridge >= 0.20).astype(int)

tabla_predicciones_ridge_test = pd.DataFrame({
    "y_true": y_test_principal.reset_index(drop=True),
    "proba_ridge": proba_test_ridge,
    "pred_ridge_umbral_050": pred_test_ridge_050,
```

```
    "pred_ridge_umbral_020": pred_test_ridge_020
})

display(tabla_predicciones_ridge_test.head())

exportar_csv(
    tabla_predicciones_ridge_test,
    "MLG-ridge_predicciones_test.csv",
    "mlg",
    "Probabilidades y clasificaciones preliminares del modelo logístico Ridge en el conjunto de
prueba."
)

# ============================================================================
# 8.2.6 EVALUACIÓN GLOBAL DEL MODELO RIDGE
# ============================================================================

from sklearn.metrics import roc_auc_score, average_precision_score, brier_score_loss

auc_train_ridge = roc_auc_score(y_train_principal, proba_train_ridge)
auc_test_ridge = roc_auc_score(y_test_principal, proba_test_ridge)
ap_test_ridge = average_precision_score(y_test_principal, proba_test_ridge)
brier_test_ridge = brier_score_loss(y_test_principal, proba_test_ridge)

tabla_metricas_globales_ridge = pd.DataFrame([{
    "modelo": "Logistic Ridge",
    "auc_train": auc_train_ridge,
    "auc_test": auc_test_ridge,
    "average_precision_test": ap_test_ridge,
    "brier_test": brier_test_ridge
}])

tabla_metricas_globales_ridge_vista = tabla_metricas_globales_ridge.copy()
for col in ["auc_train", "auc_test", "average_precision_test", "brier_test"]:
    tabla_metricas_globales_ridge_vista[col] =
tabla_metricas_globales_ridge_vista[col].round(4)

display(tabla_metricas_globales_ridge_vista)

exportar_csv(
    tabla_metricas_globales_ridge,
    "MLG-ridge_metricas_globales.csv",
    "mlg",
    "Métricas globales de desempeño del modelo logístico Ridge en entrenamiento y prueba."
)

# ============================================================================
# 8.2.7 EVALUACIÓN POR UMBRALES DE CLASIFICACIÓN
# ============================================================================

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
f1_score, roc_curve

# Umbral de Youden calculado en entrenamiento
fpr_train_ridge, tpr_train_ridge, thresholds_train_ridge = roc_curve(y_train_principal,
proba_train_ridge)
youden_index_ridge = tpr_train_ridge - fpr_train_ridge
idx_best_youden_ridge = np.argmax(youden_index_ridge)
umbral_youden_ridge = thresholds_train_ridge[idx_best_youden_ridge]
```

```
print("Umbral de Youden (train):", round(umbral_youden_ridge, 4))

# Predicciones por umbral
pred_test_ridge_youden = (proba_test_ridge >= umbral_youden_ridge).astype(int)

def calcular_metricas_clasificacion(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    sensibilidad = recall_score(y_true, y_pred)
    especificidad = tn / (tn + fp) if (tn + fp) > 0 else np.nan
    precision = precision_score(y_true, y_pred, zero_division=0)
    f1 = f1_score(y_true, y_pred, zero_division=0)
    accuracy = accuracy_score(y_true, y_pred)

    return {
        "tn": tn,
        "fp": fp,
        "fn": fn,
        "tp": tp,
        "accuracy": accuracy,
        "sensibilidad": sensibilidad,
        "especificidad": especificidad,
        "precision": precision,
        "f1_score": f1
    }

metricas_ridge_050 = calcular_metricas_clasificacion(y_test_principal, pred_test_ridge_050)
metricas_ridge_020 = calcular_metricas_clasificacion(y_test_principal, pred_test_ridge_020)
metricas_ridge_youden = calcular_metricas_clasificacion(y_test_principal,
pred_test_ridge_youden)

tabla_umbral_ridge = pd.DataFrame([
    {"umbral": 0.50, **metricas_ridge_050},
    {"umbral": 0.20, **metricas_ridge_020},
    {"umbral": umbral_youden_ridge, **metricas_ridge_youden}
])

tabla_umbral_ridge_vista = tabla_umbral_ridge.copy()
for col in ["umbral", "accuracy", "sensibilidad", "especificidad", "precision", "f1_score"]:
    tabla_umbral_ridge_vista[col] = tabla_umbral_ridge_vista[col].round(4)

display(tabla_umbral_ridge_vista)

exportar_csv(
    tabla_umbral_ridge,
    "MLG-ridge_metricas_por_umbral.csv",
    "mlg",
    "Métricas de clasificación del modelo logístico Ridge para umbrales 0.50, 0.20 y Youden."
)

# ============================================================================
# 8.2.8 MATRICES DE CONFUSIÓN DEL MODELO RIDGE
# ============================================================================

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay

fig, axes = plt.subplots(1, 3, figsize=(18, 5))
```

```
cms = [
    ("Umbral 0.50", confusion_matrix(y_test_principal, pred_test_ridge_050)),
    ("Umbral 0.20", confusion_matrix(y_test_principal, pred_test_ridge_020)),
    ("Umbral Youden", confusion_matrix(y_test_principal, pred_test_ridge_youden))
]

for ax, (titulo, cm) in zip(axes, cms):
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, ax=ax)
    ax.set_title(f"Ridge - {titulo}")
    ax.set_xlabel("Predicción")
    ax.set_ylabel("Real")

plt.tight_layout()
plt.show()

# ============================================================================
# 8.2.9 CALIBRACIÓN DEL MODELO RIDGE
# ============================================================================

from sklearn.calibration import calibration_curve

frac_pos_ridge, mean_pred_ridge = calibration_curve(
    y_test_principal,
    proba_test_ridge,
    n_bins=10,
    strategy="quantile"
)

tabla_calibracion_ridge = pd.DataFrame({
    "probabilidad_media_predicha": mean_pred_ridge,
    "fraccion_positivos_observada": frac_pos_ridge
})

tabla_calibracion_ridge_vista = tabla_calibracion_ridge.round(4)
display(tabla_calibracion_ridge_vista)

exportar_csv(
    tabla_calibracion_ridge,
    "MLG-ridge_tabla_calibracion.csv",
    "mlg",
    "Tabla de calibración del modelo logístico Ridge en el conjunto de prueba."
)

plt.figure(figsize=(6, 6))
plt.plot(mean_pred_ridge, frac_pos_ridge, marker="o", label="Ridge")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Calibración perfecta")
plt.xlabel("Probabilidad predicha")
plt.ylabel("Fracción observada de positivos")
plt.title("Curva de calibración - Logistic Ridge")
plt.legend()
plt.grid(alpha=0.3)
plt.show()


# ============================================================================
# 8.2.10 CURVA ROC DEL MODELO RIDGE
# ============================================================================
```

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr_train_ridge_roc, tpr_train_ridge_roc, _ = roc_curve(y_train_principal, proba_train_ridge)
fpr_test_ridge_roc, tpr_test_ridge_roc, _ = roc_curve(y_test_principal, proba_test_ridge)

plt.figure(figsize=(7, 6))
plt.plot(fpr_train_ridge_roc, tpr_train_ridge_roc, label=f"Train AUC = {auc_train_ridge:.4f}")
plt.plot(fpr_test_ridge_roc, tpr_test_ridge_roc, label=f"Test AUC = {auc_test_ridge:.4f}")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Azar")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Curva ROC - Logistic Ridge")
plt.legend()
plt.grid(alpha=0.3)
plt.show()

tabla_roc_ridge_test = pd.DataFrame({
    "fpr_test": fpr_test_ridge_roc,
    "tpr_test": tpr_test_ridge_roc
})

exportar_csv(
    tabla_roc_ridge_test,
    "MLG-ridge_curva_roc_test.csv",
    "mlg",
    "Puntos de la curva ROC del modelo logístico Ridge en el conjunto de prueba."
)

# ============================================================================
# 8.2.11 EXPORTACIÓN DE MATRICES DE CONFUSIÓN
# ============================================================================

tabla_cm_ridge_050 = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_ridge_050),
    index=["real_0", "real_1"],
    columns=["pred_0", "pred_1"]
)

tabla_cm_ridge_020 = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_ridge_020),
    index=["real_0", "real_1"],
    columns=["pred_0", "pred_1"]
)

tabla_cm_ridge_youden = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_ridge_youden),
    index=["real_0", "real_1"],
    columns=["pred_0", "pred_1"]
)

exportar_csv(
    tabla_cm_ridge_050,
    "MLG-ridge_matriz_confusion_umbral_050.csv",
    "mlg",
    "Matriz de confusión del modelo logístico Ridge con umbral 0.50."
)

exportar_csv(
```

```
    tabla_cm_ridge_020,
    "MLG-ridge_matriz_confusion_umbral_020.csv",
    "mlg",
    "Matriz de confusión del modelo logístico Ridge con umbral 0.20."
)

exportar_csv(
    tabla_cm_ridge_youden,
    "MLG-ridge_matriz_confusion_umbral_youden.csv",
    "mlg",
    "Matriz de confusión del modelo logístico Ridge con umbral de Youden."
)

# ============================================================================
# 8.2.12 DEFINICIÓN DEL CONJUNTO PRINCIPAL SIN AGE
# ============================================================================

variables_principal_sin_age = [
    "Gender",
    "Academic Pressure",
    "Study Satisfaction",
    "Sleep Duration",
    "Dietary Habits",
    "Work/Study Hours",
    "Financial Stress",
    "Family History of Mental Illness"
]

num_principal_sin_age = [
    "Academic Pressure",
    "Study Satisfaction",
    "Work/Study Hours",
    "Financial Stress"
]

cat_principal_sin_age = [
    "Gender",
    "Sleep Duration",
    "Dietary Habits",
    "Family History of Mental Illness"
]

X_principal_sin_age = df_clean[variables_principal_sin_age].copy()

X_train_principal_sin_age = X_principal_sin_age.loc[X_train_principal.index].copy()
X_test_principal_sin_age = X_principal_sin_age.loc[X_test_principal.index].copy()

print("X_principal_sin_age:", X_principal_sin_age.shape)
print("X_train_principal_sin_age:", X_train_principal_sin_age.shape)
print("X_test_principal_sin_age:", X_test_principal_sin_age.shape)


# ============================================================================
# 8.2.12 PIPELINE RIDGE SIN AGE Y BÚSQUEDA DE HIPERPARÁMETROS
# ============================================================================

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold

preprocess_principal_sin_age = ColumnTransformer(
    transformers=[
        ("num", StandardScaler(), num_principal_sin_age),
        ("cat", OneHotEncoder(drop="first", handle_unknown="ignore"), cat_principal_sin_age),
    ],
    remainder="drop"
)

pipe_logit_ridge_sin_age = Pipeline(steps=[
    ("preprocess", preprocess_principal_sin_age),
    ("model", LogisticRegression(
        penalty="l2",
        solver="lbfgs",
        max_iter=5000,
        random_state=random_state_modelado
    ))
])

param_grid_ridge_sin_age = {
    "model__C": np.logspace(-3, 3, 13)
}

cv_ridge_sin_age = StratifiedKFold(
    n_splits=n_splits_cv,
    shuffle=True,
    random_state=random_state_modelado
)

grid_ridge_sin_age = GridSearchCV(
    estimator=pipe_logit_ridge_sin_age,
    param_grid=param_grid_ridge_sin_age,
    scoring="roc_auc",
    cv=cv_ridge_sin_age,
    n_jobs=-1,
    refit=True,
    return_train_score=True
)

grid_ridge_sin_age.fit(X_train_principal_sin_age, y_train_principal)

print("Mejor C sin Age:", grid_ridge_sin_age.best_params_["model__C"])
print("Mejor AUC promedio CV sin Age:", round(grid_ridge_sin_age.best_score_, 4))

# ============================================================================
# 8.2.12 MÉTRICAS GLOBALES DE RIDGE SIN AGE
# ============================================================================

mejor_modelo_ridge_sin_age = grid_ridge_sin_age.best_estimator_

proba_train_ridge_sin_age =
mejor_modelo_ridge_sin_age.predict_proba(X_train_principal_sin_age)[:, 1]
proba_test_ridge_sin_age =
mejor_modelo_ridge_sin_age.predict_proba(X_test_principal_sin_age)[:, 1]

auc_train_ridge_sin_age = roc_auc_score(y_train_principal, proba_train_ridge_sin_age)
auc_test_ridge_sin_age = roc_auc_score(y_test_principal, proba_test_ridge_sin_age)
```

```
ap_test_ridge_sin_age = average_precision_score(y_test_principal, proba_test_ridge_sin_age)
brier_test_ridge_sin_age = brier_score_loss(y_test_principal, proba_test_ridge_sin_age)

tabla_metricas_ridge_sin_age = pd.DataFrame([{
    "modelo": "Logistic Ridge sin Age",
    "mejor_C": grid_ridge_sin_age.best_params_["model__C"],
    "auc_train": auc_train_ridge_sin_age,
    "auc_test": auc_test_ridge_sin_age,
    "average_precision_test": ap_test_ridge_sin_age,
    "brier_test": brier_test_ridge_sin_age,
    "brecha_auc_train_test": auc_train_ridge_sin_age - auc_test_ridge_sin_age
}])

tabla_metricas_ridge_sin_age_vista = tabla_metricas_ridge_sin_age.copy()
for col in ["mejor_C", "auc_train", "auc_test", "average_precision_test", "brier_test",
"brecha_auc_train_test"]:
    tabla_metricas_ridge_sin_age_vista[col] = tabla_metricas_ridge_sin_age_vista[col].round(4)

display(tabla_metricas_ridge_sin_age_vista)

exportar_csv(
    tabla_metricas_ridge_sin_age,
    "MLG-ridge_sin_age_metricas_globales.csv",
    "mlg",
    "Métricas globales del modelo logístico Ridge sin la variable Age."
)

# ============================================================================
# 8.2.12 COMPARACIÓN DIRECTA ENTRE RIDGE CON AGE Y RIDGE SIN AGE
# ============================================================================

tabla_comparacion_ridge_age = pd.DataFrame([
    {
        "modelo": "Ridge con Age",
        "mejor_C": grid_ridge.best_params_["model__C"],
        "auc_train": auc_train_ridge,
        "auc_test": auc_test_ridge,
        "average_precision_test": ap_test_ridge,
        "brier_test": brier_test_ridge,
        "brecha_auc_train_test": auc_train_ridge - auc_test_ridge
    },
    {
        "modelo": "Ridge sin Age",
        "mejor_C": grid_ridge_sin_age.best_params_["model__C"],
        "auc_train": auc_train_ridge_sin_age,
        "auc_test": auc_test_ridge_sin_age,
        "average_precision_test": ap_test_ridge_sin_age,
        "brier_test": brier_test_ridge_sin_age,
        "brecha_auc_train_test": auc_train_ridge_sin_age - auc_test_ridge_sin_age
    }
])

tabla_comparacion_ridge_age_vista = tabla_comparacion_ridge_age.copy()
for col in ["mejor_C", "auc_train", "auc_test", "average_precision_test", "brier_test",
"brecha_auc_train_test"]:
    tabla_comparacion_ridge_age_vista[col] = tabla_comparacion_ridge_age_vista[col].round(4)

display(tabla_comparacion_ridge_age_vista)
```

```
exportar_csv(
    tabla_comparacion_ridge_age,
    "MLG-ridge_comparacion_con_y_sin_age.csv",
    "mlg",
    "Comparación entre la regresión logística Ridge con Age y sin Age."
)
```

Código B 4. Random Forest Classifier

El Código B4 presenta el fragmento correspondiente al modelo Random Forest Classifier. Incluye la construcción del pipeline, la partición entrenamiento/prueba, la búsqueda de hiperparámetros mediante GridSearchCV, la selección de la mejor configuración, el ajuste del modelo, la estimación de probabilidades, la importancia de predictores, el cálculo de métricas globales, la curva ROC, la calibración y la evaluación por umbrales.

```
# ============================================================================
# 8.3.1 CONSTRUCCIÓN DEL PIPELINE Y BÚSQUEDA DE HIPERPARÁMETROS
# ============================================================================

from sklearn.base import clone
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder

preprocess_rf = ColumnTransformer(
    transformers=[
        ("num", "passthrough", num_principal),
        ("cat", OneHotEncoder(drop="first", handle_unknown="ignore"), cat_principal),
    ],
    remainder="drop"
)

pipe_rf = Pipeline(steps=[
    ("preprocess", preprocess_rf),
    ("model", RandomForestClassifier(
        random_state=random_state_modelado,
        n_jobs=-1
    ))
])

param_grid_rf = {
    "model__n_estimators": [300, 500],
    "model__max_depth": [None, 10, 20],
    "model__min_samples_split": [2, 10],
    "model__min_samples_leaf": [1, 5],
    "model__max_features": ["sqrt", 0.5]
}

grid_rf = GridSearchCV(
    estimator=pipe_rf,
```

```
    param_grid=param_grid_rf,
    scoring="roc_auc",
    cv=cv_ridge,
    n_jobs=-1,
    refit=True,
    return_train_score=True
)

grid_rf.fit(X_train_principal, y_train_principal)

print("Mejores hiperparámetros RF:")
print(grid_rf.best_params_)
print("Mejor AUC promedio en CV:", round(grid_rf.best_score_, 4))

# ============================================================================
# 8.3.2 RESULTADOS DE LA VALIDACIÓN CRUZADA
# ============================================================================

tabla_cv_rf = pd.DataFrame(grid_rf.cv_results_)[[
    "param_model__n_estimators",
    "param_model__max_depth",
    "param_model__min_samples_split",
    "param_model__min_samples_leaf",
    "param_model__max_features",
    "mean_train_score",
    "std_train_score",
    "mean_test_score",
    "std_test_score",
    "rank_test_score"
]].sort_values("rank_test_score").reset_index(drop=True)

tabla_cv_rf_vista = tabla_cv_rf.copy()
for col in ["mean_train_score", "std_train_score", "mean_test_score", "std_test_score"]:
    tabla_cv_rf_vista[col] = pd.to_numeric(tabla_cv_rf_vista[col]).round(4)

display(tabla_cv_rf_vista.head(10))

exportar_csv(
    tabla_cv_rf,
    "RFC-resultados_validacion_cruzada.csv",
    "rfc",
    "Resultados de validación cruzada para la selección de hiperparámetros de Random Forest."
)

tabla_best_rf = pd.DataFrame([{
    "best_n_estimators": grid_rf.best_params_["model__n_estimators"],
    "best_max_depth": grid_rf.best_params_["model__max_depth"],
    "best_min_samples_split": grid_rf.best_params_["model__min_samples_split"],
    "best_min_samples_leaf": grid_rf.best_params_["model__min_samples_leaf"],
    "best_max_features": grid_rf.best_params_["model__max_features"],
    "best_auc_cv": grid_rf.best_score_
}])

display(tabla_best_rf)

exportar_csv(
    tabla_best_rf,
    "RFC-mejores_hiperparametros.csv",
    "rfc",
```

```
    "Mejores hiperparámetros y desempeño promedio de validación cruzada de Random Forest."
)

# ============================================================================
# 8.3.3 IMPORTANCIA RELATIVA DE LOS PREDICTORES
# ============================================================================

mejor_modelo_rf = grid_rf.best_estimator_

feature_names_rf = mejor_modelo_rf.named_steps["preprocess"].get_feature_names_out()
importancias_rf = mejor_modelo_rf.named_steps["model"].feature_importances_

tabla_importancias_rf = pd.DataFrame({
    "variable_transformada": feature_names_rf,
    "importancia": importancias_rf
}).sort_values("importancia", ascending=False).reset_index(drop=True)

display(tabla_importancias_rf.head(15))

exportar_csv(
    tabla_importancias_rf,
    "RFC-importancias_variables_transformadas.csv",
    "rfc",
    "Importancia relativa de las variables transformadas en Random Forest."
)

def mapear_variable_original(nombre):
    if nombre.startswith("num__"):
        return nombre.replace("num__", "")
    if nombre.startswith("cat__"):
        resto = nombre.replace("cat__", "")
        for cat in cat_principal:
            if resto.startswith(cat + "_"):
                return cat
        return resto
    return nombre

tabla_importancias_rf["variable_original"] =
tabla_importancias_rf["variable_transformada"].apply(mapear_variable_original)

tabla_importancias_rf_agregada = (
    tabla_importancias_rf
    .groupby("variable_original", as_index=False)["importancia"]
    .sum()
    .sort_values("importancia", ascending=False)
    .reset_index(drop=True)
)

tabla_importancias_rf_agregada_vista = tabla_importancias_rf_agregada.copy()
tabla_importancias_rf_agregada_vista["importancia"] =
tabla_importancias_rf_agregada_vista["importancia"].round(4)

display(tabla_importancias_rf_agregada_vista)

exportar_csv(
    tabla_importancias_rf_agregada,
    "RFC-importancias_variables_agregadas.csv",
    "rfc",
    "Importancia relativa agregada por variable original en Random Forest."
```

```
)

# ============================================================================
# 8.3.4 GENERACIÓN DE PROBABILIDADES PREDICHAS
# ============================================================================

proba_train_rf = mejor_modelo_rf.predict_proba(X_train_principal)[:, 1]
proba_test_rf = mejor_modelo_rf.predict_proba(X_test_principal)[:, 1]

pred_test_rf_050 = (proba_test_rf >= 0.50).astype(int)
pred_test_rf_020 = (proba_test_rf >= 0.20).astype(int)

tabla_predicciones_rf_test = pd.DataFrame({
    "y_true": y_test_principal.reset_index(drop=True),
    "proba_rf": proba_test_rf,
    "pred_rf_umbral_050": pred_test_rf_050,
    "pred_rf_umbral_020": pred_test_rf_020
})

display(tabla_predicciones_rf_test.head())

exportar_csv(
    tabla_predicciones_rf_test,
    "RFC-predicciones_test.csv",
    "rfc",
    "Probabilidades y clasificaciones preliminares de Random Forest en el conjunto de prueba."
)

# ============================================================================
# 8.3.5 EVALUACIÓN GLOBAL DEL MODELO RANDOM FOREST
# ============================================================================

auc_train_rf = roc_auc_score(y_train_principal, proba_train_rf)
auc_test_rf = roc_auc_score(y_test_principal, proba_test_rf)
ap_test_rf = average_precision_score(y_test_principal, proba_test_rf)
brier_test_rf = brier_score_loss(y_test_principal, proba_test_rf)

tabla_metricas_globales_rf = pd.DataFrame([{
    "modelo": "Random Forest",
    "auc_train": auc_train_rf,
    "auc_test": auc_test_rf,
    "average_precision_test": ap_test_rf,
    "brier_test": brier_test_rf
}])

tabla_metricas_globales_rf_vista = tabla_metricas_globales_rf.copy()
for col in ["auc_train", "auc_test", "average_precision_test", "brier_test"]:
    tabla_metricas_globales_rf_vista[col] = tabla_metricas_globales_rf_vista[col].round(4)

display(tabla_metricas_globales_rf_vista)

exportar_csv(
    tabla_metricas_globales_rf,
    "RFC-metricas_globales.csv",
    "rfc",
    "Métricas globales de desempeño de Random Forest en entrenamiento y prueba."
)

# ============================================================================
```

```
# 8.3.6 EVALUACIÓN POR UMBRALES DE CLASIFICACIÓN
# ================================================================================

fpr_train_rf, tpr_train_rf, thresholds_train_rf = roc_curve(y_train_principal, proba_train_rf)
youden_index_rf = tpr_train_rf - fpr_train_rf
idx_best_youden_rf = np.argmax(youden_index_rf)
umbral_youden_rf = thresholds_train_rf[idx_best_youden_rf]

print("Umbral de Youden RF (train):", round(umbral_youden_rf, 4))

pred_test_rf_youden = (proba_test_rf >= umbral_youden_rf).astype(int)

metricas_rf_050 = calcular_metricas_clasificacion(y_test_principal, pred_test_rf_050)
metricas_rf_020 = calcular_metricas_clasificacion(y_test_principal, pred_test_rf_020)
metricas_rf_youden = calcular_metricas_clasificacion(y_test_principal, pred_test_rf_youden)

tabla_umbral_rf = pd.DataFrame([
    {"umbral": 0.50, **metricas_rf_050},
    {"umbral": 0.20, **metricas_rf_020},
    {"umbral": umbral_youden_rf, **metricas_rf_youden}
])

tabla_umbral_rf_vista = tabla_umbral_rf.copy()
for col in ["umbral", "accuracy", "sensibilidad", "especificidad", "precision", "f1_score"]:
    tabla_umbral_rf_vista[col] = tabla_umbral_rf_vista[col].round(4)

display(tabla_umbral_rf_vista)

exportar_csv(
    tabla_umbral_rf,
    "RFC-metricas_por_umbral.csv",
    "rfc",
    "Métricas de clasificación de Random Forest para umbrales 0.50, 0.20 y Youden."
)

# ================================================================================
# 8.3.7 MATRICES DE CONFUSIÓN DEL MODELO RANDOM FOREST
# ================================================================================

fig, axes = plt.subplots(1, 3, figsize=(18, 5))

cms_rf = [
    ("Umbral 0.50", confusion_matrix(y_test_principal, pred_test_rf_050)),
    ("Umbral 0.20", confusion_matrix(y_test_principal, pred_test_rf_020)),
    ("Umbral Youden", confusion_matrix(y_test_principal, pred_test_rf_youden))
]

for ax, (titulo, cm) in zip(axes, cms_rf):
    sns.heatmap(cm, annot=True, fmt="d", cmap="Greens", cbar=False, ax=ax)
    ax.set_title(f"Random Forest - {titulo}")
    ax.set_xlabel("Predicción")
    ax.set_ylabel("Real")

plt.tight_layout()
plt.show()

tabla_cm_rf_050 = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_rf_050),
    index=["real_0", "real_1"],
```

```
    columns=["pred_0", "pred_1"]
)

tabla_cm_rf_020 = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_rf_020),
    index=["real_0", "real_1"],
    columns=["pred_0", "pred_1"]
)

tabla_cm_rf_youden = pd.DataFrame(
    confusion_matrix(y_test_principal, pred_test_rf_youden),
    index=["real_0", "real_1"],
    columns=["pred_0", "pred_1"]
)

exportar_csv(tabla_cm_rf_050, "RFC-matriz_confusion_umbral_050.csv", "rfc", "Matriz de
confusión de Random Forest con umbral 0.50.")
exportar_csv(tabla_cm_rf_020, "RFC-matriz_confusion_umbral_020.csv", "rfc", "Matriz de
confusión de Random Forest con umbral 0.20.")
exportar_csv(tabla_cm_rf_youden, "RFC-matriz_confusion_umbral_youden.csv", "rfc", "Matriz de
confusión de Random Forest con umbral de Youden.")


# ============================================================================
# 8.3.8 CURVA ROC DEL MODELO RANDOM FOREST
# ============================================================================

fpr_train_rf_roc, tpr_train_rf_roc, _ = roc_curve(y_train_principal, proba_train_rf)
fpr_test_rf_roc, tpr_test_rf_roc, _ = roc_curve(y_test_principal, proba_test_rf)

plt.figure(figsize=(7, 6))
plt.plot(fpr_train_rf_roc, tpr_train_rf_roc, label=f"Train AUC = {auc_train_rf:.4f}")
plt.plot(fpr_test_rf_roc, tpr_test_rf_roc, label=f"Test AUC = {auc_test_rf:.4f}")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Azar")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Curva ROC - Random Forest")
plt.legend()
plt.grid(alpha=0.3)
plt.show()

tabla_roc_rf_test = pd.DataFrame({
    "fpr_test": fpr_test_rf_roc,
    "tpr_test": tpr_test_rf_roc
})

exportar_csv(
    tabla_roc_rf_test,
    "RFC-curva_roc_test.csv",
    "rfc",
    "Puntos de la curva ROC de Random Forest en el conjunto de prueba."
)

# ============================================================================
# 8.3.9 CALIBRACIÓN DEL MODELO RANDOM FOREST
# ============================================================================

frac_pos_rf, mean_pred_rf = calibration_curve(
    y_test_principal,
```

```
    proba_test_rf,
    n_bins=10,
    strategy="quantile"
)

tabla_calibracion_rf = pd.DataFrame({
    "probabilidad_media_predicha": mean_pred_rf,
    "fraccion_positivos_observada": frac_pos_rf
})

tabla_calibracion_rf_vista = tabla_calibracion_rf.round(4)
display(tabla_calibracion_rf_vista)

exportar_csv(
    tabla_calibracion_rf,
    "RFC-tabla_calibracion.csv",
    "rfc",
    "Tabla de calibración de Random Forest en el conjunto de prueba."
)

plt.figure(figsize=(6, 6))
plt.plot(mean_pred_rf, frac_pos_rf, marker="o", label="Random Forest")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Calibración perfecta")
plt.xlabel("Probabilidad predicha")
plt.ylabel("Fracción observada de positivos")
plt.title("Curva de calibración - Random Forest")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

Código B 5. Comparación entre modelos y análisis de robustez

El Código B5 presenta la comparación entre la regresión logística Ridge y Random Forest Classifier. Incluye la tabla comparativa de métricas, AUC en entrenamiento y prueba, Average Precision, Brier Score, brecha de AUC, curva ROC conjunta, comparación por umbrales, validación cruzada repetida, intervalos de confianza bootstrap y diferencia de AUC entre modelos.

```
# ============================================================================
# 8.4.1 TABLA COMPARATIVA DE MÉTRICAS GLOBALES
# ============================================================================

tabla_comparativa_global = pd.DataFrame([
    {
        "modelo": "Logistic Ridge",
        "auc_train": auc_train_ridge,
        "auc_test": auc_test_ridge,
        "average_precision_test": ap_test_ridge,
        "brier_test": brier_test_ridge
    },
    {
        "modelo": "Random Forest",
        "auc_train": auc_train_rf,
        "auc_test": auc_test_rf,
        "average_precision_test": ap_test_rf,
        "brier_test": brier_test_rf
    }
```

```
])

tabla_comparativa_global_vista = tabla_comparativa_global.copy()
for col in ["auc_train", "auc_test", "average_precision_test", "brier_test"]:
    tabla_comparativa_global_vista[col] = tabla_comparativa_global_vista[col].round(4)

display(tabla_comparativa_global_vista)

exportar_csv(
    tabla_comparativa_global,
    "COM-tabla_comparativa_metricas_globales.csv",
    "com",
    "Comparación de métricas globales entre la regresión logística Ridge y Random Forest."
)

# ============================================================================
# VALIDACIÓN CRUZADA REPETIDA Y ESTABILIDAD DE LAS MÉTRICAS
# ============================================================================
import numpy as np
import pandas as pd
from sklearn.base import clone
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import (
    roc_auc_score, average_precision_score, brier_score_loss, roc_curve,
    confusion_matrix, accuracy_score, precision_score, f1_score
)

# --- Configuracion de la validacion repetida ---
N_SPLITS_REP = 5
N_REPEATS_REP = 5
rkf = RepeatedStratifiedKFold(
    n_splits=N_SPLITS_REP,
    n_repeats=N_REPEATS_REP,
    random_state=RANDOM_STATE
)

# --- Insumos: SOLO conjunto de entrenamiento (evita contaminar el test) ---
X_cv = X_train_principal.reset_index(drop=True)
y_cv = pd.Series(np.asarray(y_train_principal)).reset_index(drop=True)

# --- Modelos: mejores estimadores ya seleccionados ---
modelos_cv = {
    "Logistic Ridge": grid_ridge.best_estimator_,
    "Random Forest":  grid_rf.best_estimator_,
}

def umbral_youden_desde(y_true, proba):
    """Umbral que maximiza sensibilidad + especificidad - 1 (indice de Youden)."""
    fpr, tpr, thr = roc_curve(y_true, proba)
    j = tpr - fpr
    return thr[int(np.argmax(j))]

def metricas_umbral(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred, labels=[0, 1]).ravel()
    sensibilidad = tp / (tp + fn) if (tp + fn) > 0 else np.nan   # TP/(TP+FN)
    especificidad = tn / (tn + fp) if (tn + fp) > 0 else np.nan  # TN/(TN+FP)
    return {
        "accuracy": accuracy_score(y_true, y_pred),
        "sensibilidad": sensibilidad,
```

```
        "especificidad": especificidad,
        "precision": precision_score(y_true, y_pred, zero_division=0),
        "f1_score": f1_score(y_true, y_pred, zero_division=0),
    }

# --- Bucle principal: por modelo, por particion ---
registros = []
for nombre_modelo, estimador_base in modelos_cv.items():
    for k, (idx_tr, idx_va) in enumerate(rkf.split(X_cv, y_cv)):
        repeticion = k // N_SPLITS_REP + 1
        fold = k % N_SPLITS_REP + 1

        X_tr, X_va = X_cv.iloc[idx_tr], X_cv.iloc[idx_va]
        y_tr, y_va = y_cv.iloc[idx_tr], y_cv.iloc[idx_va]

        modelo_fold = clone(estimador_base)
        modelo_fold.fit(X_tr, y_tr)

        proba_tr = modelo_fold.predict_proba(X_tr)[:, 1]
        proba_va = modelo_fold.predict_proba(X_va)[:, 1]

        # Umbral de Youden calculado SOLO en el train del fold, aplicado a su validacion
        umbral_y = umbral_youden_desde(y_tr, proba_tr)
        pred_va = (proba_va >= umbral_y).astype(int)

        fila = {
            "modelo": nombre_modelo,
            "repeticion": repeticion,
            "fold": fold,
            "auc": roc_auc_score(y_va, proba_va),
            "average_precision": average_precision_score(y_va, proba_va),
            "brier_score": brier_score_loss(y_va, proba_va),
            "umbral_youden_train": umbral_y,
        }
        fila.update(metricas_umbral(y_va, pred_va))
        registros.append(fila)

validacion_repetida_detalle = pd.DataFrame(registros)[[
    "modelo", "repeticion", "fold", "auc", "average_precision", "brier_score",
    "accuracy", "sensibilidad", "especificidad", "precision", "f1_score",
    "umbral_youden_train"
]]

print(f"Detalle: {len(validacion_repetida_detalle)} filas "
      f"({N_SPLITS_REP}x{N_REPEATS_REP} por modelo)")
display(validacion_repetida_detalle.round(4).head(12))

# --- Resumen: media, DE, mediana, percentiles, IC95% de la media ---
metricas_resumen = ["auc", "average_precision", "brier_score", "accuracy",
                    "sensibilidad", "especificidad", "precision", "f1_score"]

filas_resumen = []
for modelo in validacion_repetida_detalle["modelo"].unique():
    sub = validacion_repetida_detalle[validacion_repetida_detalle["modelo"] == modelo]
    for metrica in metricas_resumen:
        vals = sub[metrica].dropna().to_numpy()
        n = len(vals)
        media = vals.mean()
        de = vals.std(ddof=1) if n > 1 else 0.0
```

```
        margen = 1.96 * de / np.sqrt(n) if n > 0 else np.nan  # IC95% aproximado de la media
        filas_resumen.append({
            "modelo": modelo,
            "metrica": metrica,
            "media": media,
            "desviacion_estandar": de,
            "mediana": np.median(vals),
            "percentil_2_5": np.percentile(vals, 2.5),
            "percentil_97_5": np.percentile(vals, 97.5),
            "ic95_media_inferior": media - margen,
            "ic95_media_superior": media + margen,
            "n_evaluaciones": n,
        })

validacion_repetida_resumen = pd.DataFrame(filas_resumen)
cols_round = ["media", "desviacion_estandar", "mediana", "percentil_2_5",
              "percentil_97_5", "ic95_media_inferior", "ic95_media_superior"]
validacion_repetida_resumen_vista = validacion_repetida_resumen.copy()
validacion_repetida_resumen_vista[cols_round] = (
    validacion_repetida_resumen_vista[cols_round].round(4)
)
print("\nResumen (media, dispersion e incertidumbre por metrica y modelo):")
display(validacion_repetida_resumen_vista)

# --- Vista ancha: modelos en columnas (media de cada metrica) ---
try:
    resumen_ancho = validacion_repetida_resumen.pivot(
        index="metrica", columns="modelo", values="media"
    ).round(4)
    resumen_ancho_de = validacion_repetida_resumen.pivot(
        index="metrica", columns="modelo", values="desviacion_estandar"
    ).round(4)
    resumen_ancho.columns = [f"{c} (media)" for c in resumen_ancho.columns]
    resumen_ancho_de.columns = [f"{c} (DE)" for c in resumen_ancho_de.columns]
    validacion_repetida_resumen_ancho = pd.concat([resumen_ancho, resumen_ancho_de], axis=1)
    validacion_repetida_resumen_ancho =
validacion_repetida_resumen_ancho.reindex(metricas_resumen)
    print("\nResumen en formato ancho (media y DE por metrica):")
    display(validacion_repetida_resumen_ancho)
except Exception as _e:
    print("Nota: no fue posible construir la vista ancha:", repr(_e))

# --- Exportacion (seccion 'com', que si existe en SUBCARPETAS_RESULTADOS) ---
exportar_csv(
    validacion_repetida_detalle,
    "COM-validacion_repetida_detalle_metricas_completas.csv", "com",
    "Detalle por pliegue de la validacion cruzada repetida 5x5 con metricas completas."
)
exportar_csv(
    validacion_repetida_resumen,
    "COM-validacion_repetida_resumen_metricas_completas.csv", "com",
    "Resumen (media, DE, percentiles, IC95%) de la validacion cruzada repetida."
)
if "exportar_excel" in dir():
    try:
        exportar_excel(
            validacion_repetida_resumen,
            "COM-validacion_repetida_resumen_metricas_completas.xlsx", "com",
            "Resumen de la validacion cruzada repetida (Excel)."
```

```
        )
    except Exception as _e:
        print("Nota: no fue posible exportar a Excel, se conservan los CSV:", repr(_e))
else:
    print("Nota: exportar_excel no esta definida; se conservan los CSV.")

# --- Interpretacion automatica (texto dinamico segun resultados) ---
def media_de(modelo, metrica):
    f = validacion_repetida_resumen[
        (validacion_repetida_resumen["modelo"] == modelo) &
        (validacion_repetida_resumen["metrica"] == metrica)
    ]
    return float(f["media"].iloc[0]), float(f["desviacion_estandar"].iloc[0])

auc_r, auc_r_de = media_de("Logistic Ridge", "auc")
auc_f, auc_f_de = media_de("Random Forest", "auc")
brier_r, _ = media_de("Logistic Ridge", "brier_score")
brier_f, _ = media_de("Random Forest", "brier_score")
sens_r, _ = media_de("Logistic Ridge", "sensibilidad")
sens_f, _ = media_de("Random Forest", "sensibilidad")
f1_r, _ = media_de("Logistic Ridge", "f1_score")
f1_f, _ = media_de("Random Forest", "f1_score")

dif_auc = abs(auc_r - auc_f)
auc_similar = dif_auc <= (auc_r_de + auc_f_de)  # solape aproximado de dispersiones

print("\n----- INTERPRETACION AUTOMATICA -----")
print(f"AUC promedio CV: Ridge={auc_r:.4f} (DE {auc_r_de:.4f}) | "
      f"RF={auc_f:.4f} (DE {auc_f_de:.4f}).")
if auc_similar:
    print("  -> Los AUC promedio son similares: la diferencia es menor que la "
          "suma de sus desviaciones estandar (dispersiones solapadas).")
else:
    print("  -> Se observa una diferencia de AUC promedio mayor que el solape de "
          "dispersiones; aun asi, interpretar con cautela.")

if brier_r < brier_f:
    print(f"  -> Ridge mantiene mejor (menor) Brier Score: {brier_r:.4f} vs {brier_f:.4f}.")
elif brier_r > brier_f:
    print(f"  -> Random Forest presenta mejor (menor) Brier Score: {brier_f:.4f} vs
{brier_r:.4f}.")
else:
    print(f"  -> Brier Score equivalente entre modelos ({brier_r:.4f}).")

if sens_f > sens_r or f1_f > f1_r:
    print(f"  -> Random Forest mejora sensibilidad y/o F1 (Sens {sens_f:.4f} vs {sens_r:.4f}; "
          f"F1 {f1_f:.4f} vs {f1_r:.4f}).")
else:
    print(f"  -> Random Forest no mejora sensibilidad ni F1 frente a Ridge "
          f"(Sens {sens_f:.4f} vs {sens_r:.4f}; F1 {f1_f:.4f} vs {f1_r:.4f}).")

print("  -> Estas diferencias deben leerse como diferencias practicas, no como "
      "superioridad estadistica definitiva.")
print("  -> La comparacion formal de la diferencia de AUC se complementa con el "
      "análisis de bootstrap pareado.")

print("\nValidación cruzada repetida con métricas completas finalizada correctamente.")

# ============================================================================
```

```
# INTERVALOS DE CONFIANZA POR BOOTSTRAP PARA MÉTRICAS DE PRUEBA
# ============================================================================
from sklearn.metrics import roc_auc_score, average_precision_score, brier_score_loss

N_BOOT = 2000
rng = np.random.default_rng(RANDOM_STATE)

y_test_arr = np.asarray(y_test_principal)
n_test = len(y_test_arr)

def bootstrap_metricas(y_true, proba, n_boot=N_BOOT):
    aucs, aps, briers = [], [], []
    for _ in range(n_boot):
        idx = rng.integers(0, len(y_true), len(y_true))
        if len(np.unique(y_true[idx])) < 2:   # AUC indefinido si falta una clase
            continue
        aucs.append(roc_auc_score(y_true[idx], proba[idx]))
        aps.append(average_precision_score(y_true[idx], proba[idx]))
        briers.append(brier_score_loss(y_true[idx], proba[idx]))
    return np.array(aucs), np.array(aps), np.array(briers)

def fila_ic(nombre, y_true, proba):
    a, p, b = bootstrap_metricas(y_true, proba)
    def ic(v): return (np.percentile(v, 2.5), np.percentile(v, 97.5))
    return {
        "modelo": nombre,
        "AUC": round(roc_auc_score(y_true, proba), 4),
        "AUC_IC95": f"[{ic(a)[0]:.4f}, {ic(a)[1]:.4f}]",
        "AP": round(average_precision_score(y_true, proba), 4),
        "AP_IC95": f"[{ic(p)[0]:.4f}, {ic(p)[1]:.4f}]",
        "Brier": round(brier_score_loss(y_true, proba), 4),
        "Brier_IC95": f"[{ic(b)[0]:.4f}, {ic(b)[1]:.4f}]",
    }

tabla_ic_bootstrap = pd.DataFrame([
    fila_ic("Logistic Ridge", y_test_arr, np.asarray(proba_test_ridge)),
    fila_ic("Random Forest",  y_test_arr, np.asarray(proba_test_rf)),
])
print(f"Intervalos de confianza del 95% por bootstrap ({N_BOOT} remuestreos) en prueba:")
display(tabla_ic_bootstrap)

try:
    exportar_csv(tabla_ic_bootstrap,
                 "ROB-ic_bootstrap_metricas.csv", "rob",
                 "IC95% por bootstrap de AUC, AP y Brier sobre el conjunto de prueba.")
except Exception as _e:
    print("Nota: exportar_csv no disponible, se omite exportacion.")

# ============================================================================
# COMPARACIÓN PAREADA DE LA DIFERENCIA DE AUC ENTRE MODELOS
# Propósito: evaluar si la diferencia de AUC esta dentro de la variabilidad esperada
# Uso en el informe final: secciones 3.7.1 y 3.7.5
# ============================================================================
proba_r = np.asarray(proba_test_ridge)
proba_f = np.asarray(proba_test_rf)

diffs = []
for _ in range(N_BOOT):
    idx = rng.integers(0, n_test, n_test)
```

```
    if len(np.unique(y_test_arr[idx])) < 2:
        continue
    auc_r = roc_auc_score(y_test_arr[idx], proba_r[idx])
    auc_f = roc_auc_score(y_test_arr[idx], proba_f[idx])
    diffs.append(auc_r - auc_f)
diffs = np.array(diffs)

dif_obs = roc_auc_score(y_test_arr, proba_r) - roc_auc_score(y_test_arr, proba_f)
ic_low, ic_high = np.percentile(diffs, [2.5, 97.5])

tabla_dif_auc = pd.DataFrame([{
    "diferencia_AUC_ridge_menos_rf": round(dif_obs, 4),
    "IC95_inferior": round(ic_low, 4),
    "IC95_superior": round(ic_high, 4),
    "incluye_cero": bool(ic_low <= 0 <= ic_high)
}])
print("Diferencia de AUC (Ridge - Random Forest) con IC95% por bootstrap pareado:")
display(tabla_dif_auc)

try:
    exportar_csv(tabla_dif_auc,
                 "ROB-ic_diferencia_auc.csv", "rob",
                 "IC95% bootstrap pareado de la diferencia de AUC entre modelos.")
except Exception as _e:
    print("Nota: exportar_csv no disponible, se omite exportacion.")


# ============================================================================
# 8.4.2 TABLA COMPARATIVA POR UMBRALES
# ============================================================================

tabla_umbral_ridge_cmp = tabla_umbral_ridge.copy()
tabla_umbral_ridge_cmp["modelo"] = "Logistic Ridge"

tabla_umbral_rf_cmp = tabla_umbral_rf.copy()
tabla_umbral_rf_cmp["modelo"] = "Random Forest"

tabla_comparativa_umbral = pd.concat(
    [tabla_umbral_ridge_cmp, tabla_umbral_rf_cmp],
    ignore_index=True
)

tabla_comparativa_umbral = tabla_comparativa_umbral[[
    "modelo", "umbral", "tn", "fp", "fn", "tp",
    "accuracy", "sensibilidad", "especificidad", "precision", "f1_score"
]]

tabla_comparativa_umbral_vista = tabla_comparativa_umbral.copy()
for col in ["umbral", "accuracy", "sensibilidad", "especificidad", "precision", "f1_score"]:
    tabla_comparativa_umbral_vista[col] = tabla_comparativa_umbral_vista[col].round(4)

display(tabla_comparativa_umbral_vista)

exportar_csv(
    tabla_comparativa_umbral,
    "COM-tabla_comparativa_metricas_por_umbral.csv",
    "com",
    "Comparación de métricas por umbral entre la regresión logística Ridge y Random Forest."
)
```

```
# ============================================================================
# 8.4.3 COMPARACIÓN DE BRECHA TRAIN-TEST
# ============================================================================

tabla_brecha_modelos = pd.DataFrame([
    {
        "modelo": "Logistic Ridge",
        "auc_train": auc_train_ridge,
        "auc_test": auc_test_ridge,
        "brecha_auc_train_test": auc_train_ridge - auc_test_ridge
    },
    {
        "modelo": "Random Forest",
        "auc_train": auc_train_rf,
        "auc_test": auc_test_rf,
        "brecha_auc_train_test": auc_train_rf - auc_test_rf
    }
])

tabla_brecha_modelos_vista = tabla_brecha_modelos.copy()
for col in ["auc_train", "auc_test", "brecha_auc_train_test"]:
    tabla_brecha_modelos_vista[col] = tabla_brecha_modelos_vista[col].round(4)

display(tabla_brecha_modelos_vista)

exportar_csv(
    tabla_brecha_modelos,
    "COM-tabla_brecha_auc_train_test.csv",
    "com",
    "Comparación de la brecha entre AUC de entrenamiento y prueba para Ridge y Random Forest."
)

# ============================================================================
# 8.4.4 COMPARACIÓN VISUAL DE CURVAS ROC EN PRUEBA
# ============================================================================

plt.figure(figsize=(7, 6))
plt.plot(fpr_test_ridge_roc, tpr_test_ridge_roc, label=f"Logistic Ridge (AUC =
{auc_test_ridge:.4f})")
plt.plot(fpr_test_rf_roc, tpr_test_rf_roc, label=f"Random Forest (AUC = {auc_test_rf:.4f})")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray", label="Azar")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Comparación de curvas ROC en prueba")
plt.legend()
plt.grid(alpha=0.3)
plt.show()

tabla_roc_comparativa = pd.DataFrame({
    "fpr_ridge_test": pd.Series(fpr_test_ridge_roc),
    "tpr_ridge_test": pd.Series(tpr_test_ridge_roc),
    "fpr_rf_test": pd.Series(fpr_test_rf_roc),
    "tpr_rf_test": pd.Series(tpr_test_rf_roc)
})

exportar_csv(
    tabla_roc_comparativa,
    "COM-curvas_roc_test_modelos.csv",
    "com",
```

```
    "Puntos de las curvas ROC en prueba para la regresión logística Ridge y Random Forest."
)

# ============================================================================
# 8.5.1 TABLA DE ROLES METODOLÓGICOS DE LOS MODELOS
# ============================================================================

tabla_roles_modelos = pd.DataFrame([
    {
        "modelo": "Regresión logística explicativa",
        "conjunto_predictores": "Modelo principal",
        "rol_metodologico": "Interpretación de asociaciones ajustadas",
        "entra_comparacion_principal": "No",
        "comentario_clave": "Se utiliza para lectura sustantiva de coeficientes y OR, con
cautela por colinealidad residual."
    },
    {
        "modelo": "Logistic Ridge",
        "conjunto_predictores": "Modelo principal",
        "rol_metodologico": "Referencia principal de la familia logística",
        "entra_comparacion_principal": "Sí",
        "comentario_clave": "Controla mejor la colinealidad y mantiene comparabilidad con
Random Forest."
    },
    {
        "modelo": "Random Forest",
        "conjunto_predictores": "Modelo principal",
        "rol_metodologico": "Segundo modelo principal del estudio",
        "entra_comparacion_principal": "Sí",
        "comentario_clave": "Permite contrastar un algoritmo no lineal frente a la regresión
Ridge."
    },
    {
        "modelo": "Logistic Ridge Ampliado",
        "conjunto_predictores": "Modelo ampliado",
        "rol_metodologico": "Análisis complementario de incremento predictivo",
        "entra_comparacion_principal": "No",
        "comentario_clave": "Evalúa el efecto de incorporar una variable muy próxima al
desenlace."
    }
])

display(tabla_roles_modelos)

exportar_csv(
    tabla_roles_modelos,
    "COM-tabla_roles_metodologicos_modelos.csv",
    "com",
    "Resumen del papel metodológico de cada modelo dentro del bloque de modelado."
)

# ============================================================================
# 8.5.2 TABLA DE SÍNTESIS COMPARATIVA DEL DESEMPEÑO
# ============================================================================

tabla_sintesis_desempeno = pd.DataFrame([
    {
        "modelo": "Logistic Ridge",
        "tipo_modelo": "Principal",
```

```
        "auc_train": auc_train_ridge,
        "auc_test": auc_test_ridge,
        "average_precision_test": ap_test_ridge,
        "brier_test": brier_test_ridge,
        "brecha_auc_train_test": auc_train_ridge - auc_test_ridge
    },
    {
        "modelo": "Random Forest",
        "tipo_modelo": "Principal",
        "auc_train": auc_train_rf,
        "auc_test": auc_test_rf,
        "average_precision_test": ap_test_rf,
        "brier_test": brier_test_rf,
        "brecha_auc_train_test": auc_train_rf - auc_test_rf
    }
])

tabla_sintesis_desempeno_vista = tabla_sintesis_desempeno.copy()
for col in ["auc_train", "auc_test", "average_precision_test", "brier_test",
"brecha_auc_train_test"]:
    tabla_sintesis_desempeno_vista[col] = tabla_sintesis_desempeno_vista[col].round(4)

display(tabla_sintesis_desempeno_vista)

exportar_csv(
    tabla_sintesis_desempeno,
    "COM-tabla_sintesis_desempeno_modelos.csv",
    "com",
    "Síntesis comparativa de desempeño entre los modelos predictivos del estudio."
)

# ============================================================================
# 8.5.3 TABLA DE SÍNTESIS SUSTANTIVA DE HALLAZGOS
# ============================================================================

tabla_sintesis_hallazgos = pd.DataFrame([
    {
        "dimension": "Presión académica",
        "logit_explicativo": "Asociación positiva fuerte (OR = 2.3203)",
        "ridge_principal": "Coeficiente positivo más alto",
        "random_forest": "Mayor importancia relativa",
        "ridge_ampliado": "Segundo predictor más fuerte"
    },
    {
        "dimension": "Estrés financiero",
        "logit_explicativo": "Asociación positiva relevante (OR = 1.7646)",
        "ridge_principal": "Coeficiente positivo alto",
        "random_forest": "Segunda mayor importancia",
        "ridge_ampliado": "Mantiene peso predictivo alto"
    },
    {
        "dimension": "Hábitos alimenticios no saludables",
        "logit_explicativo": "Asociación positiva muy fuerte (OR = 2.9523)",
        "ridge_principal": "Uno de los coeficientes más altos",
        "random_forest": "Importancia intermedia",
        "ridge_ampliado": "Permanece entre los principales predictores"
    },
    {
        "dimension": "Edad",
```

```
        "logit_explicativo": "Asociación inversa (OR = 0.8999)",
        "ridge_principal": "Coeficiente negativo relevante",
        "random_forest": "Tercera mayor importancia",
        "ridge_ampliado": "Mantiene contribución protectora"
    },
    {
        "dimension": "Satisfacción con el estudio",
        "logit_explicativo": "Asociación inversa (OR = 0.7904)",
        "ridge_principal": "Coeficiente negativo moderado",
        "random_forest": "Importancia menor que presión y estrés",
        "ridge_ampliado": "Permanece como señal protectora"
    },
    {
        "dimension": "Menos de 5 horas de sueño",
        "logit_explicativo": "Asociación positiva (OR = 1.4616)",
        "ridge_principal": "Coeficiente positivo moderado",
        "random_forest": "Importancia baja-media",
        "ridge_ampliado": "Mantiene señal positiva"
    },
    {
        "dimension": "Antecedentes familiares",
        "logit_explicativo": "Asociación positiva (OR = 1.2716)",
        "ridge_principal": "Coeficiente positivo pequeño",
        "random_forest": "Importancia baja",
        "ridge_ampliado": "Mantiene contribución positiva"
    },
    {
        "dimension": "Ideación suicida",
        "logit_explicativo": "No incluida",
        "ridge_principal": "No incluida",
        "random_forest": "No incluida",
        "ridge_ampliado": "Predictor dominante del modelo"
    }
])

display(tabla_sintesis_hallazgos)

exportar_csv(
    tabla_sintesis_hallazgos,
    "COM-tabla_sintesis_hallazgos_sustantivos.csv",
    "com",
    "Síntesis sustantiva de los patrones más relevantes identificados en el bloque de
modelado."
)

# ============================================================================
# 8.6 PATRONES NO LINEALES IDENTIFICADOS POR RANDOM FOREST
# ============================================================================

from sklearn.inspection import PartialDependenceDisplay

variables_pdp_rf = [
    "Academic Pressure",
    "Financial Stress",
    "Age"
]

tabla_variables_pdp_rf = pd.DataFrame({
    "variable": variables_pdp_rf,
```

```
    "justificacion": [
        "Variable con mayor importancia relativa en Random Forest.",
        "Variable con alta importancia relativa y asociación relevante con la variable
objetivo.",
        "Variable sustantivamente importante y con comportamiento potencialmente no lineal."
    ]
})

display(tabla_variables_pdp_rf)

exportar_csv(
    tabla_variables_pdp_rf,
    "RFC-variables_dependencia_parcial.csv",
    "rfc",
    "Variables seleccionadas para la exploración de patrones no lineales mediante dependencia
parcial."
)

fig, ax = plt.subplots(figsize=(18, 5))

PartialDependenceDisplay.from_estimator(
    estimator=mejor_modelo_rf,
    X=X_train_principal,
    features=variables_pdp_rf,
    kind="average",
    grid_resolution=50,
    ax=ax
)

plt.suptitle("Dependencia parcial - Random Forest", y=1.02)
plt.tight_layout()
exportar_grafica(
    "RFC-dependencia_parcial_variables_principales.png",
    "rfc",
    "Gráficos de dependencia parcial para variables relevantes del modelo Random Forest."
)
plt.show()
```