# UNIVERSIDAD NACIONAL DEL SANTA
## ESCUELA DE POSGRADO
## Programa de Doctorado en Ingeniería de Sistemas e Informática

# Modelo para la predicción del éxito profesional en egresados de la UNS mediante aprendizaje automático basado en datos académicos y socioeconómicos

**Tesis para optar el grado de Doctor en Ingeniería de Sistemas e Informática**


**Autor:**

**Ms. López Heredia, Johan Max Alexander**
**Código ORCID: 0009-0003-1653-5835**

**Asesor:**

**Dr. Caselli Gismondi, Hugo Esteban**
**Código ORCID: 0000-0002-2812-6727**

**DNI N° 32819296**


**Línea de Investigación**
**Computación Aplicada**

**Nuevo Chimbote - PERÚ**
**2026**

## CONSTANCIA DE ASESORAMIENTO DE LA TESIS

Yo, Dr. Hugo Esteban Caselli Gismondi, mediante la presente certifico mi asesoramiento de la Tesis de doctorado titulada: ***"MODELO PARA LA PREDICCIÓN DEL ÉXITO PROFESIONAL EN EGRESADOS DE LA UNS MEDIANTE APRENDIZAJE AUTOMÁTICO BASADO EN DATOS ACADÉMICOS Y SOCIOECONÓMICOS",*** elaborado por el doctorando Ms. Johan Max Alexander López Heredia, para obtener el Grado Académico de Doctor en Ingeniería de Sistemas e Informática en la Escuela de Posgrado de la Universidad Nacional del Santa.

Nuevo Chimbote, marzo de 2026

.....

Dr. Hugo Esteban Caselli Gismondi
ASESOR
CODIGO ORCID 0000-0002-2812-6727
DNI Nro. 32819296

**HOJA DEL AVAL DEL JURADO EVALUADOR**

# MODELO PARA LA PREDICCIÓN DEL ÉXITO PROFESIONAL EN EGRESADOS DE LA UNS MEDIANTE APRENDIZAJE AUTOMÁTICO BASADO EN DATOS ACADÉMICOS Y SOCIOECONÓMICOS

TESIS PARA OPTAR EL GRADO DE DOCTOR EN INGENIERÍA DE SISTEMAS E INFORMÁTICA

.....

Dra. Lizbeth Dora Briones Pereyra
PRESIDENTE
CODIGO ORCID0000-0003-0626-7227
DNI N° 32960646

.....

Dr. Guillermo Edward Gil Albarrán
SECRETARIO
CODIGO ORCID 0000-0003-3782-6765
DNI N° 32960958

.....

Dr. Hugo Esteban Caselli Gismondi
VOCAL
CODIGO ORCID: 0000-0002-2812-6727
DNI N° 32819296

UNS
ESCUELA DE POSGRADO

## ACTA DE EVALUACIÓN DE SUSTENTACIÓN DE TESIS

A los diez días del mes de marzo del año 2026, siendo las 11:00 am horas, en el aula P-01 de la Escuela de Posgrado de la Universidad Nacional del Santa, se reunieron los miembros del Jurado Evaluador, designados mediante Resolución Directoral Nº 063-2025-EPG-UNS de fecha 22.01.2026, conformado por los docentes: Dra. Lizbeth Dora Briones Pereyra (Presidente), Dr. Guillermo Edward Gil Albarran (Secretario) y Dr. Hugo Esteban Caselli Gismondi (Vocal); con la finalidad de evaluar la tesis intitulada: ***"MODELO PARA LA PREDICCIÓN DEL ÉXITO EN EGRESADOS DE LA UNS MEDIANTE APRENDIZAJE AUTOMÁTICO BASADO EN DATOS ACADÉMICOS Y SOCIOECONÓMICOS";*** presentado por el tesista **Johan Max Alexander López Heredia** egresado del programa de Doctorado en Ingeniería de Sistemas e Informática.

Sustentación autorizada mediante Resolución Directoral N° 188-2026-EPG-UNS de fecha 03 de marzo de 2026.

La presidenta del jurado autorizó el inicio del acto académico; producido y concluido el acto de sustentación de tesis, los miembros del jurado procedieron a la evaluación respectiva, haciendo una serie de preguntas y recomendaciones al tesista, quien dio respuestas a las interrogantes y observaciones.

El jurado después de deliberar sobre aspectos relacionados con el trabajo, contenido y sustentación del mismo y con las sugerencias pertinentes, declara la sustentación como Aprobado, asignándole la calificación de 17.

Siendo las 12:15 horas del mismo día se da por finalizado el acto académico, firmando la presente acta en señal de conformidad.

**Dra. Lizbeth Dora Briones Pereyra**
**Presidenta**

**Dr. Guillermo Edward Gil Albarran**
**Secretario**

**Dr. Hugo Esteban Caselli Gismondi**
**Vocal/Asesor**

turnitin

# Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

| | |
|---|---|
| Autor de la entrega: | Johan Max Alexander López Heredia |
| Título del ejercicio: | Informe Final 2026 |
| Título de la entrega: | Modelo para la predicción del éxito profesional en egresados ... |
| Nombre del archivo: | ITD-JMALH-V1.14_-_Final.pdf |
| Tamaño del archivo: | 3.46M |
| Total páginas: | 138 |
| Total de palabras: | 29,260 |
| Total de caracteres: | 182,564 |
| Fecha de entrega: | 20-mar-2026 04:14p. m. (UTC-0500) |
| Identificador de la entrega: | 2682532044 |

UNIVERSIDAD NACIONAL DEL SANTA
ESCUELA DE POSGRADO
Programa de Doctorado en Ingeniería de Sistemas e Informática

Modelo para la predicción del éxito profesional en egresados de la UNS mediante aprendizaje automático basado en datos académicos y socioeconómicos

Tesis para optar el grado de Doctor en Ingeniería de Sistemas e Informática

Autor:
Ms. López Heredia, Johan Max Alexander
Código ORCID: 0009-0003-1653-5835

Asesor:
Dr. Caselli Gismondi, Hugo Esteban
Código ORCID: 0000-0002-2812-6727
DNI N° 32819296

Línea de Investigación
Computación Aplicada

Nuevo Chimbote - PERÚ
2026

Modelo para la predicción del éxito profesional en egresados de la UNS mediante aprendizaje automático basado en datos académicos y socioeconómicos

INFORME DE ORIGINALIDAD

| 14% | 12% | 6% | 10% |
|---|---|---|---|
| INDICE DE SIMILITUD | FUENTES DE INTERNET | PUBLICACIONES | TRABAJOS DEL ESTUDIANTE |

FUENTES PRIMARIAS

| | | |
|---|---|---|
| 1 | Submitted to Universitat Oberta de Catalunya<br>Trabajo del estudiante | 2% |
| 2 | ciencialatina.org<br>Fuente de Internet | 1% |
| 3 | repositorio.uns.edu.pe<br>Fuente de Internet | 1% |
| 4 | pyimagesearch.com<br>Fuente de Internet | 1% |
| 5 | Submitted to Chester College of Higher Education<br>Trabajo del estudiante | <1% |
| 6 | Submitted to Sheffield Hallam University<br>Trabajo del estudiante | <1% |
| 7 | Submitted to unibuc<br>Trabajo del estudiante | <1% |
| 8 | Submitted to ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey<br>Trabajo del estudiante | <1% |
| 9 | artemiopadilla.github.io<br>Fuente de Internet | <1% |

**Dedicatoria**

A mis padres, por su inquebrantable dedicación y respaldo incondicional, pilares fundamentales en mi desarrollo académico y personal.

A mis hermanos, por su solidaridad y apoyo constante durante los períodos de mayor exigencia a lo largo de esta travesía académica.

**Agradecimientos**

A mi asesor Dr. Hugo Caselli Gismondi, por su sapienza e inquebrantable pasión por la ciencia de datos. Su orientación y compromiso académico fueron fundamentales para la culminación exitosa de esta investigación.

A mis jurados, por su destacada sapiencia y valiosas observaciones, que contribuyeron significativamente al perfeccionamiento de este trabajo.

ÍNDICE

## ÍNDICE DE TABLAS



## ÍNDICE DE FIGURAS

**RESUMEN**

Esta investigación presenta el desarrollo de un modelo predictivo para estimar el éxito profesional de egresados de Ingeniería de Sistemas e Informática de la Universidad Nacional del Santa (UNS). Se recopilaron datos académicos (promedio de notas, ciclos cursados), socioeconómicos (dependencia económica, convivencia familiar) y laborales (tipo de contrato, certificaciones, relación con la carrera). Tras la limpieza y codificación de variables, se aplicaron técnicas de oversampling para balancear clases minoritarias y se entrenaron algoritmos de aprendizaje automático (Random Forest, XGBoost y MLPClassifier).

Los resultados evidenciaron que Random Forest y XGBoost alcanzaron precisión y macro F1-score del 100% en los conjuntos de validación y prueba, resultado que, si bien podría reflejar sobreajuste dada la muestra reducida (n=96), superó ampliamente el umbral del 80% inicialmente propuesto. Además, la importancia de variables subraya que factores como los estudios de posgrado y las certificaciones pesan más que la nota promedio en la predicción del éxito. Estos hallazgos confirman la hipótesis de que la conjunción de datos académicos y socioeconómicos permite anticipar el nivel de éxito, ofreciendo a la UNS una herramienta de diagnóstico y acción para fortalecer la inserción laboral y la formación continua de sus estudiantes.

**Palabras clave:** Aprendizaje automático, Éxito profesional, Modelo predictivo, Datos socioeconómicos, Universidad Nacional del Santa, Random Forest, Validación cruzada.

**ABSTRACT**

This study presents the development of a predictive model to estimate professional success among graduates of the Systems and Informatics Engineering program at the National University of Santa (UNS). Academic data (grade point average, course cycles), socioeconomic details (financial dependency, family environment), and labor factors (contract type, certifications, job relevance to the field) were collected. After cleaning and encoding the dataset, oversampling techniques were applied to handle minority classes, and several machine learning algorithms (Random Forest, XGBoost, and MLPClassifier) were trained.

Results showed Random Forest and XGBoost achieving 100% accuracy and macro F1-score in both validation and test sets, surpassing the initial 80% target. Variable-importance analysis revealed that postgraduate studies and certifications often outweigh grade averages in predicting success. These findings confirm the hypothesis that merging academic and socioeconomic data can effectively anticipate success levels, thus giving UNS a diagnostic tool to bolster employability strategies and enhance continuous student training.

**Keywords**: Machine learning, Professional success, Predictive model, Socioeconomic data, National University of Santa, Random Forest, Cross-validation

# I. INTRODUCCIÓN

## 1.1. Descripción

El desempeño profesional de los egresados universitarios es un tema de gran interés para las instituciones de educación superior, ya que refleja en parte la calidad de la formación impartida (Peña-Ayala, 2014). En ese sentido, la capacidad de predecir el éxito profesional de los estudiantes a partir de sus características académicas y socioeconómicas puede optimizar las estrategias educativas y de apoyo estudiantil (Kotsiantis et al., 2004). Si bien se han desarrollado algunos modelos predictivos para estimar el rendimiento académico, son escasos los enfocados en el éxito profesional posterior al egreso (Hung et al., 2012). Entre las técnicas utilizadas destacan los árboles de decisión, redes neuronales y algoritmos de clasificación supervisada (Osmanbekov et al., 2020). No obstante, aún no se dispone de un modelo predictivo específico para los egresados de la UNS, que integre variables académicas y socioeconómicas (Lopez Sánchez et al., 2019).

Varios estudios previos han aplicado técnicas de aprendizaje automático para predecir resultados académicos y laborales de estudiantes universitarios. En Nigeria, Baffa et al. (2023) desarrollaron modelos predictivos basados en regresión logística, árboles de decisión y bosques aleatorios para estimar la empleabilidad de estudiantes antes de graduarse, utilizando predictores como el promedio académico, prácticas profesionales y actividades extracurriculares. El mejor desempeño se obtuvo con el bosque aleatorio, que alcanzó una precisión del 98%. Los autores concluyeron que más datos estudiantiles permitirían mejorar los modelos y predecir la empleabilidad con mayor certeza.

En India, Bhagavan et al. (2020) desarrollaron un algoritmo híbrido HLVQ combinando LVQ y AdaBoost para predecir la probabilidad de graduación y empleo de estudiantes a partir de sus calificaciones e indicadores académicos. Los experimentos mostraron que HLVQ superaba en precisión y eficiencia a métodos como redes neuronales, regresión logística y árboles aleatorios.

En Colombia, Bedoya et al. (2019) desarrollaron un modelo predictivo aplicando técnicas de minería de datos para identificar los factores

socioculturales que influyen en el tiempo que tardan los egresados universitarios en conseguir su primer empleo. El estudio con egresados de Ingeniería de Sistemas halló que el nivel educativo de la madre, el rendimiento académico y el factor de ingreso eran predictores significativos del tiempo hasta el primer empleo.

En Colombia, Talero (2023) desarrolló una herramienta predictiva aplicando árboles de decisión, regresión logística y otros algoritmos de aprendizaje automático para estimar la probabilidad de retiro de los afiliados a la Asociación de Egresados de la Universidad de Los Andes. El modelo encontró que las variables más influyentes eran la cantidad de facturas pagadas y en mora, meses de antigüedad y familiares inscritos.

En el contexto peruano, Alarcón et al. (2022) propusieron un modelo predictivo para procesar grandes volúmenes de datos académicos en una universidad nacional. El modelo contempla cuatro dimensiones: soporte tecnológico, analítica de negocio, analítica de datos y decisiones basadas en datos; apuntando a mejorar el tratamiento de los registros académicos a través de técnicas como minería de datos.

La gestión del desempeño académico de los estudiantes universitarios es un desafío que las instituciones educativas enfrentan desde hace mucho tiempo. En el contexto peruano, Caselli Gismondi (2021) desarrolló un modelo predictivo aplicando técnicas de machine learning y deep learning para contribuir a mejorar el seguimiento a estudiantes universitarios. Específicamente, Caselli entrenó y evaluó modelos de redes neuronales para predecir si los estudiantes completarían sus estudios, se titularían o abandonarían la carrera, usando datos académicos y socioeconómicos de estudiantes de 4 escuelas de ingeniería de la UNS. El mejor modelo obtuvo un 98.97% de precisión en el conjunto de entrenamiento, demostrando su potencial para identificar tempranamente

## 1.2. Formulación del Problema

En la UNS, la diversidad de especialidades es amplia, con algunas de ellas mostrando un número significativamente mayor de egresados en comparación con otras. Por ejemplo, la especialidad de Enfermería lidera la lista con un 10.58% del total de egresados, seguida de cerca por Ingeniería Agroindustrial y Ingeniería Civil, de un total de 10,424 egresados. Estas carreras destacan por su relevancia y demanda dentro del contexto educativo de la universidad.

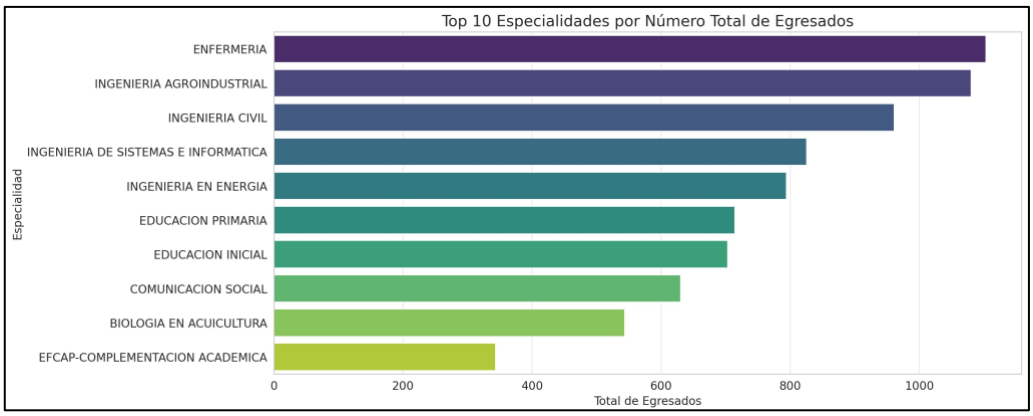


*Figura 1: Top 10 especialidades por número total de egresados*

La Figura 1 muestra las 10 principales especialidades por número de egresados de un total de 10,424. La Escuela de Enfermería tiene el mayor número con 1,113 egresados (10.56%). Ingeniería de Sistemas e Informática se ubica en cuarto lugar con 825 egresados (7.91%). Las otras especialidades en los primeros puestos son Ingeniería Agroindustrial e Ingeniería Civil, con entre 987 y 902 egresados cada una (9.47% - 8.66%).

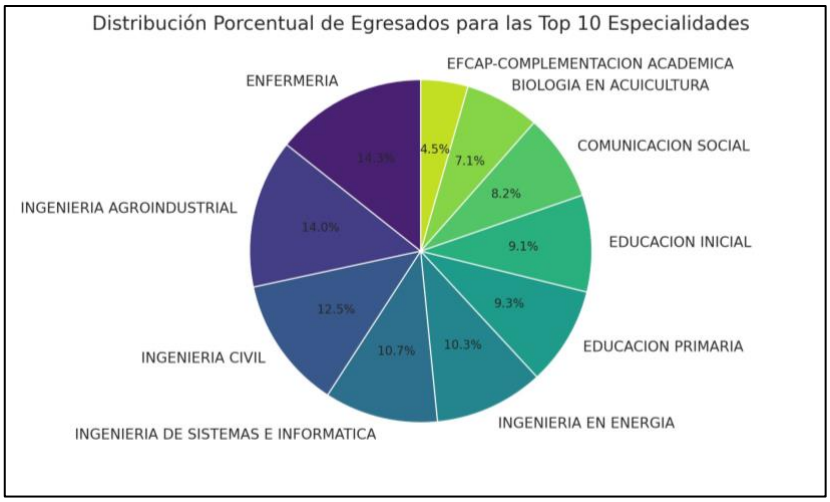


*Figura 2: Distribución porcentual de egresados para las Top 10 especialidades*

En la figura 2, se presenta una distribución porcentual de los egresados para las 10 principales especialidades, lo que permite verificar cómo se distribuyen los egresados entre estas especialidades.

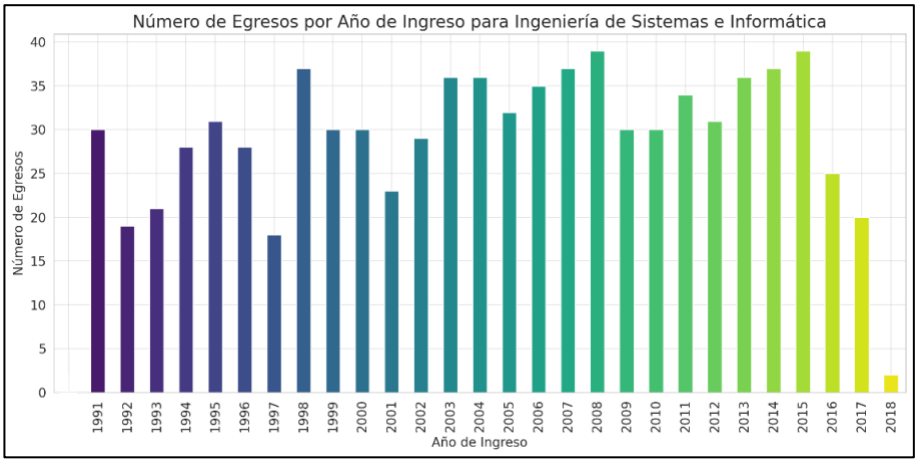


*Figura 3: Número de egresos por año de ingreso para Ingeniería de Sistemas e Informática*

En la figura 3, podemos observar una tendencia general en el número de egresos a medida que avanzan los años, con algunas fluctuaciones.

La formulación del problema se sustentó en la necesidad imperiosa de comprender, de manera cuantitativa y cualitativa, la relación existente entre el éxito profesional de los egresados y sus antecedentes académicos y socioeconómicos. Tras un análisis de la literatura y la evaluación de estudios previos, se evidenció que, si bien existen modelos predictivos aplicados a

contextos internacionales, en la UNS no se contaba con una herramienta específica que integrara de forma sistemática estas variables para anticipar el desempeño profesional posterior a la graduación.

Ante este escenario, se planteó el interrogante central que orientó la investigación:

**¿Cómo predecir el éxito profesional de los egresados de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNS mediante la integración de datos académicos y socioeconómicos, para contribuir a la optimización de las estrategias educativas y de apoyo estudiantil?**

Como parte del preámbulo de esta problemática se reconoció que el éxito profesional –definido en términos de estabilidad laboral, adecuación del empleo a la formación recibida y reconocimiento en el ámbito profesional– dependía no solo de las calificaciones y el rendimiento académico, sino también de factores inherentes a las condiciones socioeconómicas en las que se formaron los estudiantes. Se constató que las oportunidades de inserción en el mercado laboral y la capacidad para alcanzar niveles superiores de desempeño estaban estrechamente ligadas a variables como el nivel de ingresos familiares, la calidad del entorno de estudio y la disponibilidad de recursos educativos.

El análisis mostró que actualmente el seguimiento del desempeño profesional de los egresados se realiza de manera manual, con procesos lentos y datos dispersos que no permiten identificar a tiempo los factores que influyen en el éxito laboral. Esta situación limita la capacidad de la institución para implementar estrategias de apoyo temprano y programas personalizados. Por ello, la investigación se orientó a desarrollar un modelo predictivo que alcance alta precisión y que además permita identificar claramente qué factores académicos y socioeconómicos son determinantes en el éxito profesional de los egresados.

En síntesis, la formulación del problema se estructuró en torno a la identificación e integración de variables determinantes, con el objetivo de

construir un modelo predictivo que permitiera, de manera efectiva, predecir el desempeño profesional a partir de los antecedentes académicos y socioeconómicos de los egresados, contribuyendo así a la mejora de las estrategias educativas y al fortalecimiento del apoyo institucional en la UNS.

## 1.3. Objetivos

### Objetivo General

Desarrollar un modelo predictivo basado en Aprendizaje Automático para estimar el grado de correlación entre el éxito profesional de los egresados de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNS y sus características académicas y socioeconómicas, con el propósito de contribuir a la optimización de las estrategias educativas y de apoyo estudiantil.

### Objetivos Específicos

1. Recolectar y analizar de manera integral los datos académicos y socioeconómicos de los egresados, estableciendo mediante el análisis estadístico aquellas variables que incidieron de forma determinante en su desempeño profesional.
2. Diseñar e implementar un modelo predictivo basado en algoritmos de aprendizaje automático que integre variables identificadas para estimar el éxito profesional
3. Validar la eficacia y precisión del modelo comparando las predicciones generadas con los datos reales de desempeño profesional, aplicando técnicas de validación cruzada y análisis estadístico.
4. Formular implicaciones y proponer recomendaciones para la mejora en la estrategia educativa y de apoyo institucional, a partir de los hallazgos obtenidos, permitiendo optimizar la intervención en la inserción laboral de los egresados.

## 1.4. Formulación de la hipótesis

*Un modelo de aprendizaje automático, basado en datos académicos y socioeconómicos, predice el éxito profesional de los egresados de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNS con un F1-Score superior a 0.80, cuyo desempeño brindaría entendimiento para optimizar las*

*actuales estrategias educativas y programas de apoyo estudiantil de la institución.*

## 1.5. Justificación e importancia

El presente trabajo se centra en crear un modelo predictivo para evaluar el éxito futuro de los egresados de la UNS, basado en datos académicos y socioeconómicos. Este modelo, apoyado en técnicas de aprendizaje automático, aspira a descubrir correlaciones entre estas variables y el desempeño posuniversitario, en consonancia con la Ley N° 30220, Ley Universitaria, que insta a las universidades a evaluar el desempeño de sus egresados para elevar la calidad educativa.

Así mismo, se considera que este trabajo es crucial para mejorar las estrategias educativas y de apoyo estudiantil de la universidad, y está alineado con el Plan Nacional de Educación Superior y Técnico-Productiva 2021-2025 (MINEDU, 2021), que promueve una educación superior inclusiva y equitativa. Los hallazgos podrían apoyar a políticas educativas y contribuir a diseñar programas y servicios de apoyo estudiantil más efectivos, cumpliendo así con los reglamentos peruanos y las metas nacionales de educación. Además, el modelo podría servir como un sistema de detección temprana para reconocer estudiantes con alto potencial de éxito profesional. Esto permitiría brindarles mentoría y oportunidades especializadas para potenciar su talento desde etapas iniciales, contribuyendo a formar profesionales de alto nivel.

# II. MARCO TEÓRICO

## 2.1. Antecedentes

La educación superior se ha convertido en un elemento decisivo para el desarrollo socioeconómico de las naciones, al formar profesionales capaces de contribuir al progreso científico y tecnológico (Peña-Ayala, 2014). En las últimas décadas, se ha observado una marcada transformación en el sistema universitario, caracterizada por la expansión de la matrícula, la diversificación de las carreras y la necesidad de garantizar estándares de calidad que respondan a las demandas de un entorno laboral cada vez más competitivo (Benavides et al., 2015). Ante esta realidad, las instituciones de educación superior han dirigido sus esfuerzos a analizar y comprender los factores que inciden en el desempeño académico y profesional de sus egresados, lo que ha dado lugar a un creciente interés por las técnicas de aprendizaje automático aplicadas a la predicción del éxito profesional (Kotsiantis et al., 2004).

En particular, el éxito profesional se entiende como la capacidad de los egresados de insertarse de manera satisfactoria en el mercado laboral, alineando su formación académica con las demandas del entorno, alcanzando estabilidad y crecimiento en sus puestos de trabajo y, en muchos casos, innovando y aportando valor a la sociedad (Baquero Pérez & Ruesga, 2019; Lavado et al., 2014). En el contexto peruano, la Universidad Nacional del Santa (UNS) se ha visto en la necesidad de profundizar en el estudio de los factores que determinan dicho éxito, atendiendo a las condiciones socioeconómicas de sus estudiantes y a la calidad de la formación impartida (UNS, 2017; UNS, 2022). Con tal propósito, se han ido adoptando progresivamente metodologías analíticas que permitan identificar, entre otras cosas, la correlación entre el rendimiento académico, las condiciones socioeconómicas y la inserción laboral (Guerrero et al., 2014; Yamada et al., 2013).

### 2.1.1. Aplicaciones de aprendizaje automático para la predicción del éxito profesional

El auge de la ciencia de datos y la inteligencia artificial ha facilitado la creación de modelos predictivos capaces de procesar grandes volúmenes de información, identificando patrones y relaciones que no son evidentes mediante métodos estadísticos convencionales (Sandoval, 2018; Murphy, 2022). Dentro de esta gama de técnicas, los algoritmos de clasificación supervisada (por ejemplo, regresión logística, árboles de decisión, bosques aleatorios) y los métodos de aprendizaje no supervisado (por ejemplo, K-means, KNN) se han utilizado en múltiples contextos educativos, desde la predicción de deserción académica hasta la proyección de la empleabilidad (Hung et al., 2012; Osmanbekov et al., 2020).

En países como Nigeria, se han implementado modelos predictivos basados en regresión logística y árboles de decisión para estimar la empleabilidad de los estudiantes antes de graduarse, incorporando variables como el promedio académico, las prácticas profesionales y la participación en actividades extracurriculares (Baffa et al., 2023). Por su parte, en India, Bhagavan et al. (2020) combinaron técnicas de aprendizaje supervisado para predecir la probabilidad de graduación y la posterior obtención de empleo, evidenciando la eficacia de integrar distintos algoritmos (por ejemplo, LVQ, AdaBoost) para mejorar la precisión de las predicciones. Estos hallazgos resaltaron la necesidad de personalizar los modelos según la disponibilidad y la calidad de los datos en cada institución.

En Latinoamérica, varios trabajos han enfatizado la importancia de las variables socioculturales y económicas en la transición de la universidad al mercado laboral. Por ejemplo, Bedoya et al. (2019) identificaron que el nivel educativo de la madre y el ingreso familiar inciden en el tiempo que tardan los egresados en conseguir su primer empleo, mientras que Talero (2023) demostró que los factores vinculados a la situación financiera y el entorno familiar influyen en la retención de afiliados a asociaciones de egresados. En conjunto, estas experiencias indican que, más allá de los logros

académicos, es fundamental contemplar las condiciones contextuales que pueden favorecer o dificultar la inserción laboral de los graduados.

### 2.1.2. Perspectiva peruana y la situación de la Universidad Nacional del Santa

El sistema universitario peruano ha enfrentado el desafío de pasar de un modelo elitista a uno masificado, lo que conlleva exigencias en términos de equidad, pertinencia y eficiencia (Benavides et al., 2015). En la actualidad, muchas universidades públicas, incluida la UNS, experimentan limitaciones presupuestales y de infraestructura, así como la necesidad de articular mejor sus planes de estudio con las demandas de un mercado laboral en constante cambio (León, 2007). Bajo esta óptica, la incorporación de metodologías analíticas basadas en aprendizaje automático representa una alternativa atractiva para aprovechar los datos que las instituciones recogen de sus estudiantes, a fin de anticipar sus necesidades, reforzar las políticas de acompañamiento y, en última instancia, mejorar la calidad de la formación.

Entre los antecedentes de la UNS en el uso de técnicas predictivas destaca el trabajo de Caselli Gismondi (2021), quien aplicó algoritmos de aprendizaje automático y deep learning para pronosticar la continuidad o el abandono de los estudiantes en carreras de ingeniería, obteniendo altos niveles de precisión en la identificación de quienes estaban en riesgo de deserción. Estos resultados, si bien centrados en la permanencia académica, abrieron la puerta para que la universidad explorara la posibilidad de extender la predicción al éxito profesional de sus egresados. A su vez, Alarcón García et al. (2022) resaltaron la importancia de reforzar la infraestructura tecnológica y la cultura de gestión de datos, de modo que se puedan integrar, de manera efectiva, los registros académicos y las encuestas socioeconómicas en un solo repositorio.

### 2.1.3. Factores determinantes del éxito profesional en egresados

De acuerdo con Baquero Pérez y Ruesga (2019), los indicadores que definen el éxito profesional de los egresados incluyen la obtención de un empleo estable, el nivel salarial, la adecuación del puesto a la formación recibida, la jornada laboral y el grado de satisfacción con el trabajo desempeñado. Sin embargo, en el contexto peruano, dichos factores se amplían al considerar la capacidad de los egresados para innovar, adaptarse a las nuevas tecnologías y contribuir al desarrollo local. Aunado a ello, el nivel de redes de contacto y la proactividad en la búsqueda de oportunidades suelen desempeñar un papel significativo (Lavado et al., 2014).

Por otra parte, la realidad socioeconómica de cada estudiante puede condicionar sus posibilidades de dedicarse tiempo completo al estudio, acceder a materiales y recursos de calidad, o participar en actividades extracurriculares que fortalezcan su perfil (Guerrero et al., 2014). Al integrar estas variables en un modelo predictivo, se espera una mayor precisión para explicar las diferencias en el desempeño profesional de los egresados (Sandoval, 2018). De hecho, diversos autores han coincidido en que la calidad de los datos y la representatividad de las muestras son esenciales para desarrollar herramientas analíticas con alto valor explicativo (Baffa et al., 2023; Osmanbekov et al., 2020).

### 2.1.4. Desafíos y oportunidades de los modelos predictivos en la educación superior

La aplicación de técnicas de aprendizaje automático para la predicción del éxito profesional presenta oportunidades, pero también retos significativos. Uno de los principales consiste en la interpretabilidad de los resultados. Aunque algoritmos complejos (por ejemplo, redes neuronales profundas) pueden alcanzar altos niveles de precisión, es crucial que los tomadores de decisiones —autoridades académicas, docentes y responsables de las áreas de seguimiento a egresados— comprendan los factores que subyacen a las predicciones (Peña-Ayala, 2014). De esta manera, se pueden diseñar políticas y programas de intervención más acertados, alineados con las verdaderas necesidades de la población estudiantil (Hung et al., 2012).

Además, la calidad y disponibilidad de los datos sigue siendo un tema prioritario. Varias instituciones no cuentan con repositorios centralizados o con registros históricos completos, lo que dificulta la elaboración de modelos robustos (Alarcón García et al., 2022). Para enfrentar este obstáculo, se han propuesto estrategias que incluyen la digitalización de documentos, la integración de sistemas de gestión académica y el uso de encuestas online, con la finalidad de captar información actualizada sobre la trayectoria académica y la situación socioeconómica de los estudiantes y egresados (Guerrero et al., 2014).

Por último, la transferibilidad de los modelos predictivos a otros contextos universitarios es un punto de debate (Osmanbekov et al., 2020). Si bien la evidencia sugiere que la mayoría de algoritmos se adaptan relativamente bien a diferentes instituciones, cada universidad presenta particularidades culturales, económicas y académicas que pueden alterar la relevancia de determinadas variables o modificar la precisión de los modelos. Por consiguiente, se requiere un esfuerzo constante de validación y refinamiento de los enfoques propuestos, así como la adopción de metodologías colaborativas que permitan compartir experiencias y buenas prácticas (Bhagavan et al., 2020).

### 2.1.5. Proyección hacia la investigación

En conclusión, los antecedentes indican que las técnicas de aprendizaje automático han evolucionado hasta convertirse en un recurso valioso para anticipar el éxito profesional de los egresados universitarios, combinando variables académicas y socioeconómicas. Las experiencias desarrolladas en distintos países —incluyendo el Perú— ponen de manifiesto el potencial de estas metodologías para orientar la planificación educativa y fortalecer la vinculación con el mercado laboral (Baffa et al., 2023; Bedoya et al., 2019). No obstante, queda claro que cada institución debe adecuar los modelos a su realidad particular, considerando la calidad de los datos disponibles, la heterogeneidad de la población estudiantil y los objetivos de mejora que se persiguen (Caselli Gismondi, 2021; UNS, 2022).

Para la Universidad Nacional del Santa, y específicamente para la Escuela Profesional de Ingeniería de Sistemas e Informática, contar con un modelo predictivo que estime el grado de correlación entre las características académicas y socioeconómicas de sus egresados y su posterior desempeño laboral constituye una estrategia fundamental para optimizar las estrategias educativas y los programas de apoyo estudiantil. La evidencia revisada respalda la pertinencia de adoptar enfoques que trasciendan el simple análisis descriptivo, incorporando métodos avanzados de minería de datos y aprendizaje automático. Este marco teórico y empírico sienta las bases para la propuesta investigativa, la cual busca no solo profundizar en la comprensión de los factores que influyen en el éxito profesional, sino también generar herramientas concretas para la toma de decisiones y la mejora continua de la formación universitaria.

## 2.2. Marco Conceptual

### 2.2.1. Modelo

Un modelo se entiende como una representación simplificada de una realidad compleja, que se utiliza como punto de referencia para analizar y predecir comportamientos futuros. En este contexto, el modelo es un esquema teórico que integra datos académicos y socioeconómicos para estimar el éxito profesional de los egresados (Real Academia Española, 2019; Hastie, Tibshirani, & Friedman, 2009).

Es pertinente distinguir entre algoritmo y modelo en el contexto del aprendizaje automático. Un algoritmo (e.g., Random Forest, XGBoost, MLPClassifier) es el procedimiento o conjunto de reglas computacionales que define cómo aprender patrones a partir de los datos. En cambio, un modelo es el resultado de aplicar un algoritmo a un conjunto de datos específico: es la representación matemática entrenada que contiene los parámetros aprendidos (pesos, árboles de decisión, umbrales) y que se utiliza para realizar predicciones sobre nuevos datos (Hastie et al., 2009; Murphy, 2022). En esta investigación, el modelo predictivo es el producto final del entrenamiento de los tres algoritmos seleccionados sobre el dataset de egresados de la EPISI, siendo Random Forest el algoritmo que generó el modelo con mejor desempeño.

### 2.2.2. Modelamiento predictivo

El modelamiento predictivo es el proceso sistemático de utilizar técnicas estadísticas y algoritmos de aprendizaje automático para pronosticar resultados futuros basados en datos históricos. Este enfoque permite identificar patrones y relaciones subyacentes entre variables, facilitando la toma de decisiones informadas (Berea, 2017).

### 2.2.3. Éxito profesional

Se define como la capacidad de un egresado para insertarse de manera efectiva en el mercado laboral, alcanzando estabilidad y reconocimiento en su campo, y demostrando que la formación recibida es pertinente para su desempeño profesional. Este concepto abarca indicadores como la obtención de empleo, el nivel salarial y la adecuación del puesto a la formación (Baquero Pérez & Ruesga, 2019).

### 2.2.4. Desempeño académico

El desempeño académico, o rendimiento educativo, se refiere al nivel de logros obtenidos por un estudiante a lo largo de su formación, medido a través de indicadores como calificaciones, tasas de aprobación y la capacidad para avanzar en su carrera sin repeticiones significativas (Salvador Blanco & Garcia-Valcarcel Muñoz-Repiso, 1989).

### 2.2.5. Seguimiento académico

El seguimiento académico consiste en la acción de monitorear de forma continua el progreso y el rendimiento de los estudiantes, permitiendo detectar tempranamente dificultades o áreas de mejora. Este proceso es fundamental para diseñar estrategias de intervención que aseguren el éxito en la formación y posterior inserción laboral (Real Academia Española, 2019).

### 2.2.6. Variables académicas

Son aquellos indicadores que reflejan el rendimiento y los logros obtenidos durante la formación universitaria. Entre ellos se incluyen el promedio de calificaciones, la tasa de aprobación, la repetición de cursos y la participación en actividades extracurriculares, que en conjunto permiten

evaluar el nivel de compromiso y eficacia del proceso educativo (Peña-Ayala, 2014).

### 2.2.7. Variables socioeconómicas

Estas variables describen el contexto económico y social de los estudiantes, tales como el ingreso familiar, el nivel educativo de los padres, las condiciones de vivienda y el acceso a recursos tecnológicos. Dichas variables influyen significativamente en la capacidad de los estudiantes para aprovechar las oportunidades académicas y, en consecuencia, su desempeño profesional (Lavado et al., 2014).

### 2.2.8. Aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial que desarrolla algoritmos capaces de aprender de los datos, identificar patrones y realizar predicciones sin una programación explícita para cada tarea. Su aplicación es esencial en el modelamiento predictivo, ya que permite procesar grandes volúmenes de información y extraer relaciones no evidentes mediante métodos tradicionales (Murphy, 2022).

### 2.2.9. Minería de datos

La minería de datos implica el proceso de explorar grandes conjuntos de datos para descubrir patrones, correlaciones y tendencias relevantes. Esta técnica complementa el aprendizaje automático al proporcionar una base de información estructurada a partir de datos complejos y heterogéneos (Peña-Ayala, 2014).

### 2.2.10. Algoritmos de clasificación supervisada

Estos algoritmos se entrenan utilizando datos etiquetados para asignar nuevas instancias a categorías predefinidas. En el contexto de este estudio, métodos como la regresión logística, árboles de decisión y bosques aleatorios se utilizan para clasificar a los egresados según su probabilidad de éxito profesional (Kotsiantis et al., 2004).

### 2.2.11. Validación cruzada

La validación cruzada es una técnica de evaluación que consiste en dividir el conjunto de datos en varios subconjuntos. Cada uno se utiliza alternativamente para entrenar y validar el modelo, lo que permite estimar su capacidad de generalización y evitar el sobreajuste (Hastie, Tibshirani, & Friedman, 2009).

### 2.2.12. Optimización de hiperparámetros

Este proceso consiste en ajustar de manera automática o manual los parámetros que controlan el comportamiento de los algoritmos de aprendizaje automático. La optimización de hiperparámetros es crucial para maximizar la precisión y el rendimiento del modelo, adaptándolo específicamente a la problemática en estudio (Feurer & Hutter, 2019).

### 2.2.13. Preprocesamiento de datos

El preprocesamiento es el conjunto de técnicas destinadas a limpiar, transformar y estructurar los datos brutos, de modo que se asegure su calidad y consistencia para el análisis. Este paso es fundamental para garantizar que el modelo predictivo opere sobre información precisa y representativa (Müller & Guido, 2016).

### 2.2.14. Interpretabilidad de modelos

La interpretabilidad se refiere a la capacidad de comprender y explicar cómo un modelo de aprendizaje automático llega a sus predicciones. Este aspecto es fundamental en aplicaciones donde las decisiones basadas en el modelo deben ser transparentes y comprensibles para los gestores educativos y otros tomadores de decisiones (Peña-Ayala, 2014; Hung et al., 2012).

### 2.2.15. Evaluación del modelo

La evaluación del modelo implica medir su desempeño utilizando métricas específicas como precisión, sensibilidad, especificidad. El desempeño de cada modelo se puede evaluar utilizando métricas apropiadas para clasificación multiclase. Este proceso permite determinar la efectividad del

modelo predictivo y su capacidad para generalizar a datos no vistos, asegurando así su aplicabilidad en contextos reales (Murphy, 2022).

# III. METODOLOGÍA

## 3.1. Enfoque de investigación

El presente estudio adoptó un enfoque cuantitativo, ya que se basa en la recolección, procesamiento y análisis de datos numéricos y categóricos mediante técnicas estadísticas y algoritmos de aprendizaje automático. Este enfoque permitió establecer relaciones entre variables académicas, socioeconómicas y el éxito profesional de los egresados, con el objetivo de desarrollar un modelo predictivo basado en evidencia empírica.

## 3.2. Método

El método empleado es de tipo correlacional, ya que busca establecer y cuantificar la relación entre las características académicas y socioeconómicas de los egresados de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNS y su éxito profesional posterior. Este método permitió identificar patrones y correlaciones entre las variables predictoras y la variable objetivo, facilitando el desarrollo de un modelo que estima el grado de éxito profesional alcanzado.

## 3.3. Diseño de investigación

La investigación se fundamenta en un diseño correlacional no experimental de enfoque cuantitativo. Este diseño se caracteriza por la ausencia de manipulación deliberada de variables, analizando las relaciones entre ellas tal como se presentan en la realidad. La aplicación de técnicas de aprendizaje automático permitió desarrollar y validar un modelo predictivo que integra múltiples dimensiones del desempeño académico y el contexto socioeconómico de los egresados.

El diseño contempló las siguientes etapas metodológicas:

1. Recopilación de datos históricos y actuales de fuentes institucionales y complementarias.
2. Integración y preprocesamiento del dataset para garantizar su calidad y consistencia.
3. Selección y entrenamiento de algoritmos de aprendizaje automático.
4. Evaluación y validación del modelo mediante métricas estadísticas apropiadas.
5. Análisis de la importancia de variables para identificar factores determinantes del éxito profesional.

## 3.4. Población

La población del estudio está conformada por los *825 egresados de la Escuela Profesional de Ingeniería de Sistemas e Informática de la Universidad Nacional del Santa (UNS)*, registrados en el sistema de gestión académica institucional. Esta población abarca egresados de diferentes cohortes y representa la totalidad de profesionales formados en esta especialidad que han completado sus estudios universitarios en la UNS.

## 3.5. Muestra

La muestra está constituida por **96 egresados** de la Escuela Profesional de Ingeniería de Sistemas e Informática de la UNS, seleccionados en función de la disponibilidad y completitud de sus datos académicos, socioeconómicos y laborales.

**Criterios de inclusión:**

- Egresados cuyos datos académicos y socioeconómicos se encuentran completos y accesibles en los registros institucionales.
- Participantes que respondieron a los instrumentos complementarios de recolección de datos (cuestionarios y entrevistas).
- Registros con información suficiente sobre su situación laboral post-egreso.

**Criterios de exclusión:**

- Egresados con datos incompletos o inconsistentes en variables críticas.
- Registros con más del 90% de valores faltantes en las variables de interés.

La muestra fue seleccionada considerando la representatividad de diferentes años de egreso, garantizando que el modelo predictivo pueda capturar variabilidad temporal en las trayectorias profesionales.

## 3.6. Operacionalización de variables

La operacionalización de variables permite definir de manera precisa cómo se midieron las características académicas, socioeconómicas y el éxito profesional de los egresados. Las variables del estudio se clasificaron de la siguiente manera:

**Variables Independientes (Predictoras):**

- **Variables Académicas:** Promedio de notas, ciclos cursados, obtención de grado de bachiller y título profesional, entre otras.
- **Variables Socioeconómicas:** Tipo de convivencia, dependencia económica, condición laboral del estudiante y del responsable económico, hábitos alimenticios y de actividad física.

**Variable Dependiente:**

- **Éxito Profesional:** Categorizada en niveles del 0 al 5, basada en la autoevaluación del egresado respecto a su inserción laboral, estabilidad, satisfacción, logros y reconocimientos profesionales.

Las variables categóricas fueron codificadas mediante técnicas de One-Hot Encoding y Label Encoding, mientras que las variables numéricas fueron normalizadas para su análisis mediante algoritmos de aprendizaje automático. La tabla completa de operacionalización se detalla en la Tabla 1.

***Tabla 1:*** Operacionalización de variables

| Variable | Definición conceptual | Indicadores | Tipo | Técnica | Instrumento |
|---|---|---|---|---|---|
| VI: Variables Académicas | Indicadores que reflejan el rendimiento y logros obtenidos durante la formación universitaria. | - Promedio de notas<br>- Ciclos cursados<br>- Grado de bachiller<br>- Título profesional | Cuantitativa | Análisis documental | Registros del sistema de gestión académica |
| VI: Variables Socioeconómicas | Características que describen el contexto económico y social de los estudiantes. | - Tipo de convivencia<br>- Dependencia económica<br>- Condición laboral<br>- Estado civil de padres | Cualitativa | Encuesta, análisis documental | Cuestionarios, registros de bienestar universitario |
| VD: Éxito Profesional | Nivel de logro alcanzado por los egresados en el ámbito laboral. | - Estabilidad laboral<br>- Salario<br>- Pertinencia del empleo<br>- Satisfacción laboral | Cuantitativa Cualitativa | Análisis estadístico, encuestas | Cuestionarios, registros laborales |

## 3.7. Técnicas e instrumentos de recolección de datos

### 3.7.1. Fuentes de datos

El análisis se basó en un dataset final obtenido en formato CSV, que integra información proveniente de diversas fuentes internas de la institución y de instrumentos de recolección primaria. Por razones de ética y confidencialidad, todos los datos fueron anonimizados para proteger la identidad de los participantes. Las principales fuentes son:

- **Datos Académicos:**

  Se extrajeron registros del sistema de gestión académica de la UNS, que incluyen variables como el código del estudiante, promedio de notas, ciclo máximo cursado, indicación de egreso, grado de bachiller y obtención del título profesional, entre otras.

- **Datos Socioeconómicos:**

  Se obtuvo información relativa al entorno familiar y condiciones personales de los estudiantes, tales como el tipo de convivencia, condiciones laborales tanto del alumno como de su responsable económico, dependencia económica, estado civil de los padres, y hábitos alimenticios y de actividad física, que permiten inferir el contexto socioeconómico.

- **Datos de Egresados:**

  Se incorporaron variables que recogen la situación laboral post-egreso, tales como si el egresado trabaja actualmente, motivos para no trabajar, tipo de organización, cargo, área de desempeño, tipo de contrato, nivel de ingresos, relación del empleo con la carrera, tiempo para conseguir el primer empleo, estudios de posgrado, certificaciones, logros, satisfacción, entre otros.

### 3.7.2. Instrumentos y Herramientas

Se utilizaron diversos instrumentos y plataformas para la recolección, procesamiento y análisis de los datos:

- **Instrumentos de Recolección de Datos:**

  **Cuestionarios estructurados y entrevistas semiestructuradas**: Aplicados a una muestra representativa de egresados para

complementar la información extraída de los registros institucionales. Estos instrumentos fueron validados mediante pruebas piloto para asegurar su claridad y pertinencia.

- **Software y Plataformas:**
  - **Lenguajes de programación**: Python y R se utilizaron para la manipulación, preprocesamiento y análisis de datos, haciendo uso de librerías como pandas, NumPy, scikit-learn y matplotlib (Müller & Guido, 2016; Murphy, 2022).
  - **Plataformas en la nube**: Google Colab y Kaggle facilitaron el procesamiento de grandes volúmenes de datos y la ejecución de algoritmos de aprendizaje automático.
  - **Repositorios de datos**: OpenML se empleó para compartir y comparar modelos predictivos y conjuntos de datos.
  - **Editores de Código**: Para ejecutar los script en local se utilizó Visual Studio Code y PyCharm versión Comunitaria.
- **Hardware:**

  Se utilizaron computadoras con procesadores modernos (por ejemplo, Intel Core i7 o superior), con al menos 16 GB de memoria RAM y unidades SSD de 512 GB, lo que garantizó un procesamiento eficiente de la información.

## 3.8. Técnicas de análisis de resultados

### 3.8.1. Análisis del dataset

El dataset final se compone de numerosas variables que han sido organizadas en tres bloques temáticos, los cuales se resumen a continuación:

**Data Académica**

Contiene las variables relacionadas con el desempeño y la trayectoria educativa. Las principales cabeceras identificadas son:

***Tabla 2:*** *Data académica*

| Variable | Descripción |
|---|---|
| estudiante.codigo | Código único asignado a cada estudiante. |
| estudiante.promedio_nota | Promedio de notas acumulado durante la carrera. |
| estudiante.max_ciclo | Número máximo de ciclo o semestre cursado. |
| estudiante.Egresado | Indicador de si el estudiante ha egresado. |
| estudiante.Bachiller | Indicador de obtención del grado de bachiller. |
| estudiante.titulo | Indicador de obtención del título profesional. |
| estudiante.TipoEstudiosRealizados | Modalidad o tipo de estudios realizados. |
| estudiante.TiempoInterrupcionEstudios | Tiempo de interrupción en la formación, en caso de haberla. |
| estudiante.TecnicaPresentacion | Técnica o modalidad de presentación de resultados académicos. |
| estudiante.RecursosApoyo | Recursos de apoyo institucional utilizados durante la formación. |
| estudiante.RazonesDesercion | Motivos que pueden haber influido en la deserción académica. |
| estudiante.RazonesEleccionCarrera | Factores que influyeron en la elección de la carrera. |
| estudiante.RazonesEleccionEstudio | Razones que determinaron la continuidad de estudios universitarios. |

## Data de Egresados

Recopila información sobre la inserción y desempeño profesional de los egresados:

***Tabla 3:*** *Data de egresados*

| Variable | Descripción |
|---|---|
| egresado.Actualmente_trabaja | Indicador de si el egresado se encuentra empleado en la actualidad. |
| egresado.Motivo_no_trabaja | Razón principal de no estar trabajando (cuando aplica). |
| egresado.Tipo_organizacion | Tipo de organización o empresa en la que se desempeña el egresado (pública, privada, etc.). |
| egresado.Cargo_puesto | Cargo o puesto que ocupa actualmente. |
| egresado.Area_desempeno | Área o sector laboral del egresado. |
| egresado.Tiempo_en_el_puesto | Tiempo que el egresado ha permanecido en su puesto actual, indicador de estabilidad laboral. |
| egresado.Tipo_contrato | Modalidad de contratación laboral (tiempo completo, parcial, temporal, etc.). |
| egresado.Nivel_ingresos | Rango de ingresos mensuales del egresado. |
| egresado.Relacion_con_carrera | Grado de relación entre el empleo actual y la formación universitaria recibida. |
| egresado.Meses_para_primer_empleo | Tiempo transcurrido desde la graduación hasta obtener el primer empleo. |
| egresado.Estudios_posgrado | Indicador de realización de estudios de posgrado o especialización. |

| | |
|---|---|
| egresado.Grado_certificacion_obtenida | Nivel de certificación obtenido en estudios complementarios. |
| egresado.Certificaciones_profesionales | Registro de certificaciones profesionales obtenidas. |
| egresado.Capacitacion_frecuente | Frecuencia de participación en actividades de capacitación o actualización profesional. |
| egresado.Idiomas | Idiomas dominados por el egresado. |
| egresado.Exito_profesional | Autoevaluación del éxito profesional del egresado. |
| egresado.Logros_profesionales | Principales logros o hitos alcanzados en su trayectoria profesional. |
| egresado.Satisfaccion_actual | Nivel de satisfacción con la situación laboral actual. |
| egresado.Cambio_empleo_12_meses | Indicador de cambios de empleo en los últimos 12 meses. |
| egresado.Reconocimientos_laborales | Registro de premios o reconocimientos obtenidos en el ámbito profesional. |
| egresado.Emprendimiento_propio | Indicador de si el egresado ha iniciado un emprendimiento propio. |
| egresado.Sector_empresa | Sector o industria en la que opera la empresa del egresado. |
| egresado.Tamanio_empresa | Tamaño de la empresa (micro, pequeña, mediana o grande). |
| egresado.Puesto_liderazgo | Indicador de si el egresado ocupa un puesto de liderazgo o gerencial. |
| egresado.Formacion_UNS_competencias | Evaluación de las competencias adquiridas durante la formación en la UNS y su incidencia en el desempeño profesional. |

### 3.8.2. Integración y preprocesamiento del dataset

Posterior a la identificación y clasificación de las variables, se procedió a integrar los datos provenientes de las diferentes fuentes mediante identificadores únicos (por ejemplo, estudiante.codigo y CODIGO_MATRICULA). El preprocesamiento incluyó:

- **Verificación de Consistencia**: Se aseguraron vínculos correctos entre registros y se corrigieron inconsistencias.
- **Limpieza y Transformación**: Se aplicaron técnicas de imputación para completar datos faltantes, normalización de variables numéricas y codificación one-hot para variables categóricas.
- **Consolidación del Dataset**: Se unificaron los datos en un único conjunto que facilita el análisis integral de la trayectoria académica, el contexto socioeconómico y la situación profesional.

## 3.9. Métodos

La metodología adoptada se basa en un diseño correlacional de enfoque cuantitativo y en la aplicación de técnicas de aprendizaje automático para el

desarrollo y validación del modelo predictivo. Los procedimientos se detallan en los siguientes apartados.

### 3.9.1. Recolección de Datos

La recolección de datos se efectuó en dos etapas:

- **Extracción de Datos Institucionales:**
  Se accedió a los sistemas de gestión académica y de bienestar de la UNS para extraer registros históricos y actualizados relacionados con el desempeño académico, las condiciones socioeconómicas y la información de egreso.

- **Aplicación de Instrumentos Complementarios:**
  Se administraron cuestionarios estructurados y se realizaron entrevistas semiestructuradas a una muestra representativa de egresados para obtener información que no estaba presente en los registros institucionales, tales como percepciones sobre el éxito profesional, logros, satisfacción y aspectos cualitativos de la inserción laboral.

### 3.9.2. Preparación y preprocesamiento de Datos

Para garantizar la calidad y homogeneidad de los datos se siguieron los siguientes pasos:

- **Limpieza de Datos:**
  Se identificaron y corrigieron valores atípicos, registros incompletos e inconsistencias en las distintas fuentes de información.

- **Transformación y Codificación:**
  Se normalizaron las variables numéricas y se codificaron las variables categóricas utilizando técnicas como la codificación one-hot, lo que facilitó su manejo en los algoritmos de aprendizaje automático.

- **Reducción de Dimensionalidad:**

  Se evaluó la aplicación de técnicas como el Análisis de Componentes Principales (PCA) para reducir la redundancia y minimizar la multicolinealidad, mejorando la eficiencia del modelo.

- **División del Dataset:**

  El conjunto de datos fue dividido en subconjuntos de entrenamiento (70%), validación (15%) y prueba (15%), lo que permitió ajustar y evaluar la capacidad de generalización del modelo.

### 3.9.3. Desarrollo del Modelo Predictivo

El modelo predictivo se desarrolló siguiendo una secuencia lógica que incluyó:

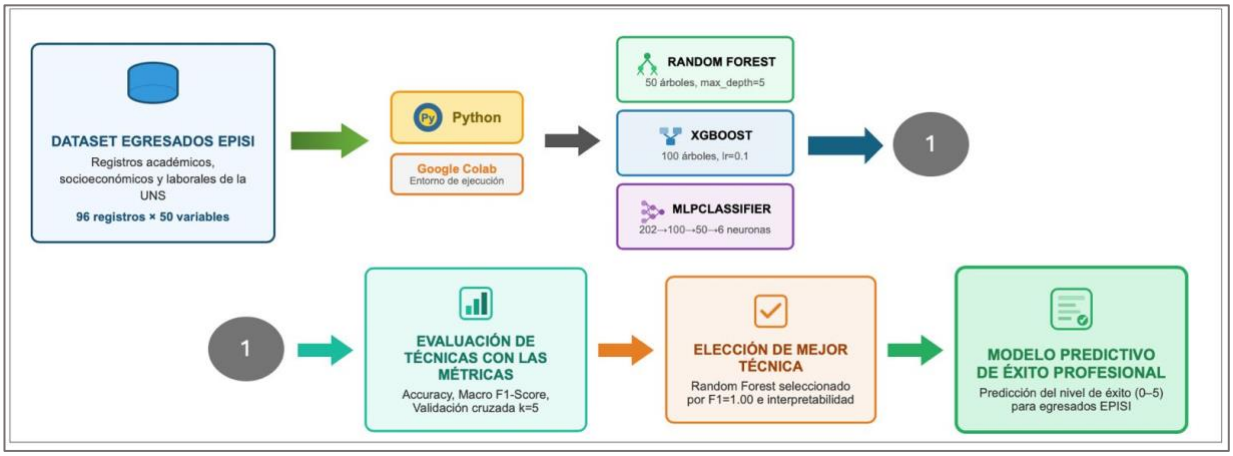


*Figura 4 - Flujo del Desarrollo del Modelo Predictivo*

- **Selección de Algoritmos:**

  Se evaluaron diversas técnicas de aprendizaje automático (regresión logística, árboles de decisión, bosques aleatorios y, en menor medida, redes neuronales) basándose en el análisis exploratorio de las variables y en la necesidad de interpretabilidad del modelo.

- **Entrenamiento del Modelo:**

  Los algoritmos seleccionados fueron entrenados utilizando el conjunto de datos de entrenamiento. Se realizó un ajuste de

hiperparámetros mediante el uso del subconjunto de validación, aplicando técnicas de validación cruzada (k-fold cross-validation) para evitar el sobreajuste.

- **Evaluación del Modelo:**
  El desempeño de cada modelo se evaluó en el conjunto de prueba utilizando métricas apropiadas para clasificación multiclase: precisión (accuracy), recall, F1-score macro y matriz de confusión. Estos indicadores permitieron seleccionar el modelo que mejor equilibró precisión y capacidad de generalización.

- **Análisis de Importancia de Variables:**
  Se realizó un análisis de las variables predictoras para identificar cuáles tenían mayor incidencia en el éxito profesional, proporcionando información clave para la optimización de las estrategias educativas y de apoyo institucional.

### 3.9.4. Validación y Contrastación de hipótesis

Para contrastar la hipótesis —que un modelo basado en datos académicos y socioeconómicos predice eficazmente el éxito profesional de los egresados— se siguió el siguiente procedimiento:

- **Comparación con Datos Reales:**
  Se compararon las predicciones generadas por el modelo con los datos reales de desempeño profesional, evaluando la precisión del modelo.

- **Criterio de Validación:**
  Se consideró que la hipótesis se confirmaba si el modelo alcanzaba un **macro F1-score superior al 80%** y si el análisis de la importancia de variables permitía identificar de manera clara los factores determinantes en el éxito profesional.

- **Interpretación de Resultados:**

  Los resultados se analizaron en detalle para extraer insights sobre la correlación entre las variables y su impacto en la inserción laboral, lo que a su vez sirvió para validar el enfoque metodológico y orientar futuras estrategias de intervención.

# IV. RESULTADOS Y DISCUSIÓN

El presente capítulo expone los resultados obtenidos en cada fase del modelo predictivo diseñado para esta investigación. La estructura se organiza en correspondencia directa con los cuatro objetivos específicos planteados, de modo que cada sección principal aborda y da respuesta a uno de ellos. La Figura 28 presenta el diseño general del modelo, mostrando las fases del proceso metodológico y su alineación con cada objetivo específico.

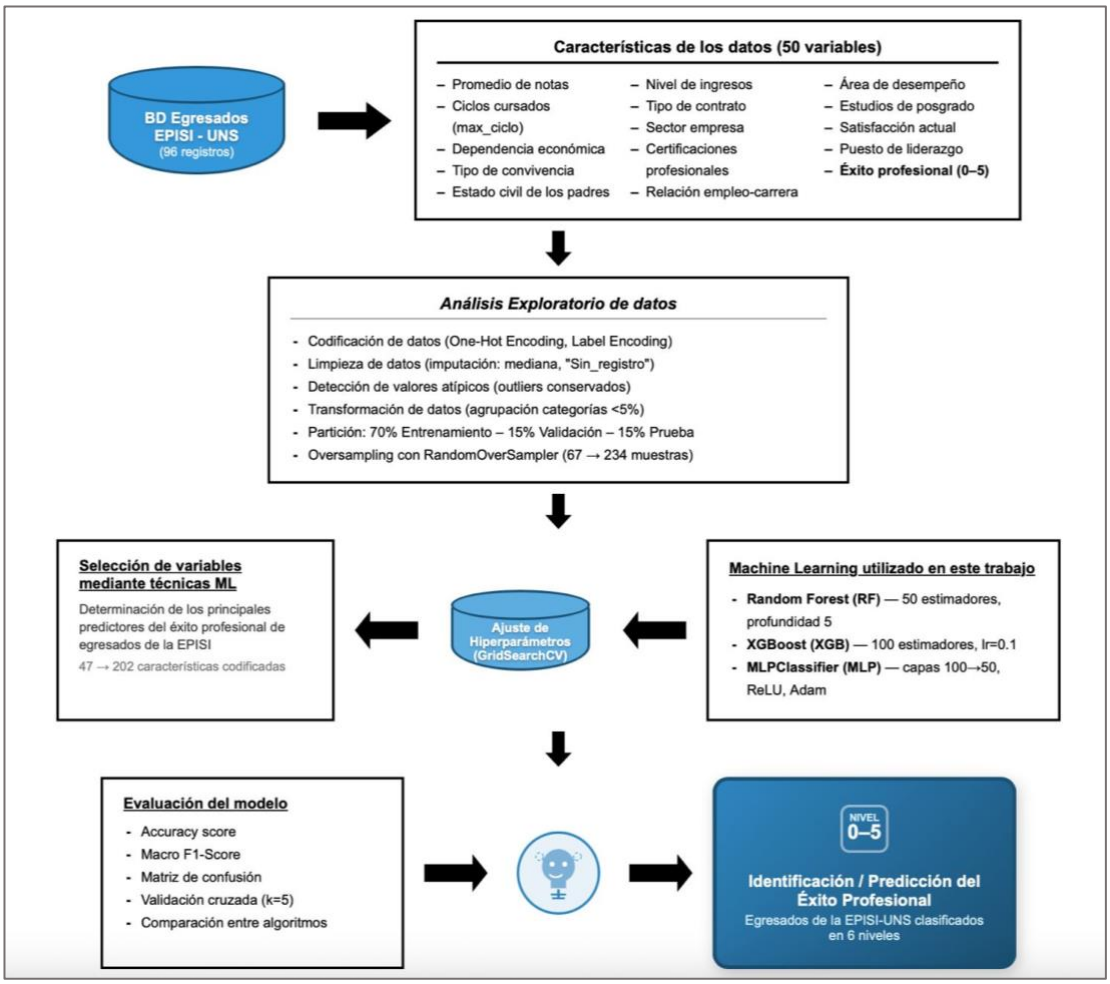


*Figura 5: Modelo Predictivo para la Predicción del Éxito Profesional*

## 4.1. Recolección y análisis integral de datos académicos y socioeconómicos

En esta sección se da respuesta al primer objetivo específico: «Recolectar y analizar de manera integral los datos académicos y socioeconómicos de los egresados, estableciendo mediante el análisis estadístico aquellas variables que incidieron de forma determinante en su desempeño profesional». Para ello, se describen las etapas de revisión del dataset, limpieza e imputación, codificación de variables categóricas y análisis exploratorio de los datos.

### 4.1.1. Revisión General de los Datos y Diagnóstico inicial del Dataset

#### 4.1.1.1. Lectura y verificación del archivo

Para la carga de los datos se implementó el script *1_data_loading.py* (en Anexo 1), que se ejecutó correctamente, generando dos tipos de resultado: (1) información básica del dataset (en Anexo 1) y (2) reportes almacenados en los directorios de resultados. El fragmento de código inicia definiendo las rutas absolutas para los directorios de **datos crudos** (*data/raw*) y **resultados** (*results/tables)* y *results/figures*). Seguidamente, se emplea la librería *pandas* para leer el archivo *dataset_final_final.csv*, el cual contiene información académica, socioeconómica y de egresados.

El script desplegó la siguiente información:

- **Número de registros**: *96*.
- **Número de columnas**: 50, de las cuales 3 son numéricas y 2 son enteras, mientras que el resto (45) son variables de tipo *object* (categóricas o texto).
- **Memoria utilizada**: aproximadamente 37.6 KB.

Esta verificación inicial confirma que la tabla mantiene consistencia en la cantidad de columnas esperadas (50 atributos), con nombres de columnas alineados a la estructura académica y de egresados descrita en capítulos anteriores.

#### 4.1.1.2. Estructura y tipos de datos

La clase del DataFrame se identificó como *pandas.core.frame.DataFrame*, confirmando que el archivo CSV se leyó en un formato adecuado para manipulación en Python. Se evidenció que los campos *estudiante.promedio_nota* y *egresado.Satisfaccion_actual* son de tipo float64, mientras que *estudiante.max_ciclo* y *estudiante.codigo* se manejan como int64. El resto de atributos —en su mayoría— son categóricos, lo que sugiere la necesidad de aplicar técnicas de codificación posterior (One-Hot Encoding o Label Encoding), conforme se avance en el preprocesamiento y se busque entrenar el modelo predictivo.

#### 4.1.1.3. Estadísticas descriptivas globales

Como parte del análisis exploratorio inicial, se calcularon estadísticas descriptivas para las variables numéricas del dataset, con el propósito de caracterizar la distribución de los datos y detectar posibles anomalías antes del modelado predictivo. Los resultados más relevantes fueron los siguientes:

- *estudiante.promedio_nota*: Promedio de 14.12, con un valor mínimo de 12.16 y máximo de 19.41. Esta distribución evidencia que los egresados encuestados mantuvieron un rendimiento académico por encima del mínimo aprobatorio (11.00 en el sistema vigesimal peruano), lo cual es coherente con su condición de egresados. La variabilidad del promedio será analizada posteriormente en relación con su poder predictivo sobre el éxito profesional (Sección 4.2.2).
- *estudiante.max_ciclo*: Valor constante en 10, lo que concuerda con la malla curricular de la EPISI (10 ciclos académicos). Al tratarse de una muestra conformada exclusivamente por egresados, este valor uniforme es esperado y confirma la integridad de los datos institucionales.
- *egresado.Satisfaccion_actual*: Media de 4.22 (en una escala Likert de 1 a 5, donde 1 representa la menor satisfacción y 5 la máxima satisfacción) y un mínimo de 3.00, lo que indica una tendencia favorable en la percepción de satisfacción entre los egresados encuestados.

Si bien no todas las columnas del dataset son numéricas, las estadísticas descriptivas presentadas confirman la variabilidad existente en el rendimiento académico (*promedio_nota*) y la presencia de una medida autorreportada de satisfacción laboral.

#### 4.1.1.4. Valores faltantes

El reporte de valores nulos (archivo *missing_values.csv*) indica que 33 de las 50 columnas presentan ausencias. Entre las variables con mayor carencia de información destacan:

- *estudiante.TiempoInterrupcionEstudios* (96 valores faltantes de 96), lo que sugiere que ninguno de los registros cuenta con este dato.
- Varias columnas ligadas a condiciones socioeconómicas y motivos de deserción, cuya presencia en el dataset se debe a la estructura del sistema de gestión académica de la UNS, que registra este campo para todos los estudiantes indistintamente, (*estudiante.RazonesDesercion*,

*estudiante.RazonesEleccionCarrera*, etc.) registran 78 valores faltantes.

- *estudiante.CondicionTrabajoAlumno* y *egresado.Tamanio_empresa* poseen 93 valores nulos cada una.

El alto número de ausencias en determinadas variables indica que se requerirá un tratamiento cuidadoso, ya sea mediante eliminación de columnas no representativas, imputación estratégica o un enfoque de ingeniería de características para no sacrificar información crítica.

#### 4.1.1.5. Distribución de la variable *Promedio notas*

Con la finalidad de ilustrar la **distribución del rendimiento académico**, se generó un histograma de *estudiante.promedio_nota* (*Figura 6*). La mayor frecuencia se concentra entre 12.5 y 14.5, pero existen algunos casos que alcanzan notas mayores a 18, mostrando una cola derecha (right-skewed).

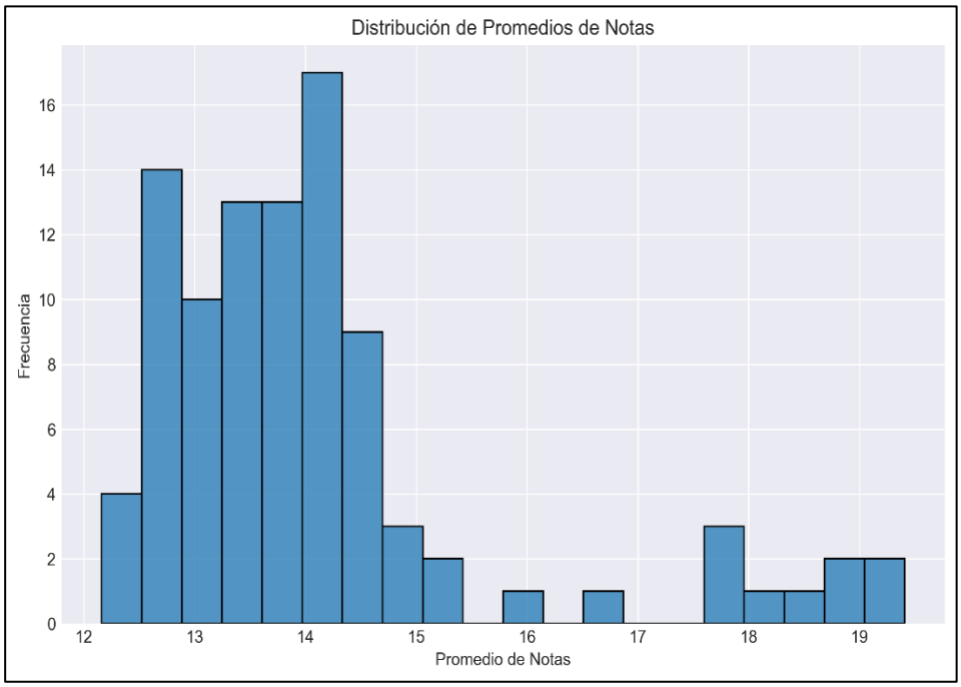


*Figura 6: Distribución de Promedios de Notas*

Este comportamiento sugiere una dispersión moderada en las calificaciones, coherente con la media de 14.12 detectada en el análisis descriptivo.

Dado esta fase en la revisión de los datos del dataset podemos hacer algunos comentarios preliminares como:

- **Coherencia general**: El dataset posee la estructura esperada, combinando datos académicos y socioeconómicos con variables propias del egresado (tipo de contrato, área de desempeño, nivel de ingresos, etc.).
- **Ausencia notable de datos**: Algunas columnas críticas tienen elevados valores nulos. Este hallazgo condiciona la estrategia de limpieza e imputación en etapas posteriores.
- **Necesidad de codificación**: La preponderancia de variables categóricas requerirá un manejo cuidadoso (p. ej., one-hot encoding), especialmente para construir el modelo de predicción sin perder información valiosa.
- **Viabilidad de análisis**: Las 96 observaciones pueden considerarse un conjunto de datos manejable; sin embargo, la baja completitud en ciertos atributos plantea retos que se abordarán en la siguiente fase de preprocesamiento.

### 4.1.2. Limpieza e imputación de los Datos

#### 4.1.2.1. Análisis de valores faltantes

La primera tarea fue cuantificar el **porcentaje de valores nulos** en cada columna. El programa generó el archivo *porcentaje_valores_faltantes.csv* y la gráfica **Top 10 - Porcentaje de Valores Faltantes por Columna** (*Figura 7*). Dentro de los hallazgos más notables destacan:

- *estudiante.TiempoInterrupcionEstudios:* **100%** de celdas vacías, lo que impidió cualquier imputación y justificó su eliminación.
- *estudiante.CondicionTrabajoAlumno* y *egresado.Tamanio_empresa*: **96.88%** de ausencia cada una, resultando muy complejas de rescatar sin introducir sesgos.
- Otras variables, como *estudiante.DondeCome* y *estudiante.RecursosApoyo*, presentaron alrededor de **81%** de faltantes.

Estos resultados evidencian la necesidad de un criterio estricto para descartar atributos con más del **90% de datos nulos**, evitando sobrecargar el dataset con información incompleta o poco confiable.

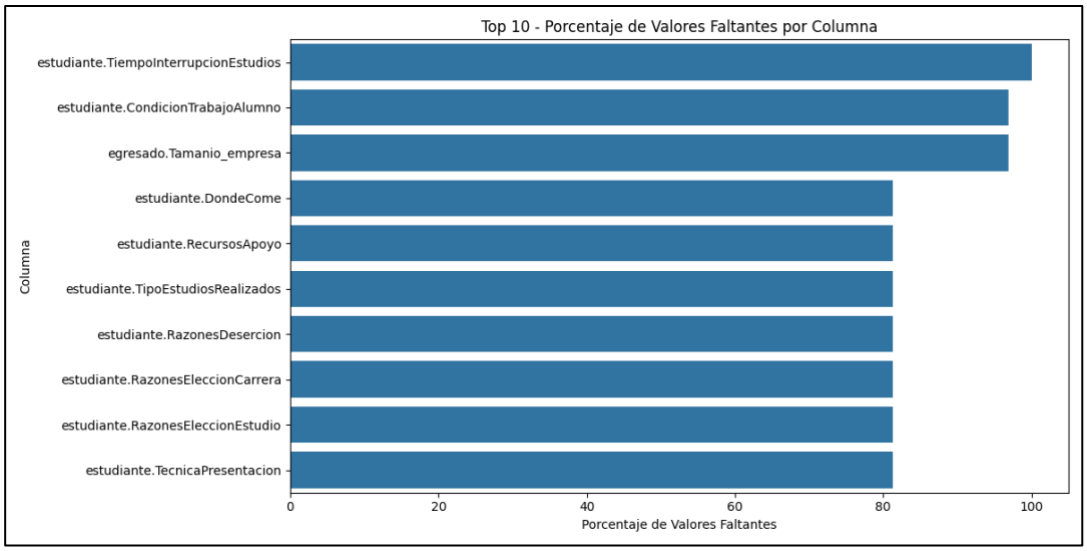


*Figura 7: Top 10 - Porcentaje de Valores Faltantes por Columna*

#### 4.1.2.2. Eliminación de columnas con más del 90% de nulos

Siguiendo el enfoque planteado en la literatura (Müller & Guido, 2016), se optó por **descartar** aquellas columnas cuyo porcentaje de faltantes superara el umbral del 90%. En este caso, fueron suprimidos:

- *estudiante.TiempoInterrupcionEstudios*
- *estudiante.CondicionTrabajoAlumno*
- *egresado.Tamanio_empresa*

Esta acción redujo la dimensionalidad del dataset de 50 a 47 columnas. Si bien se pierde la posibilidad de analizar dichas variables, la magnitud de su ausencia hacía inviable su conservación sin introducir sesgos graves.

#### 4.1.2.3. Imputación de valores en variables restantes

Una vez suprimidos los atributos más problemáticos, se aplicaron dos estrategias de imputación:

- **Imputación categórica:**

  Se reemplazaron los valores nulos de las columnas de tipo object (categóricas) por la etiqueta genérica "*Sin_registro*". Esta

aproximación evita la exclusión de filas y conserva la dimensión del dataset, además de resaltar la categoría como “desconocida” para un futuro análisis diferencial.

- **Imputación numérica:**
  Para los campos numéricos (*float64 y int64*) con pocos valores ausentes (p.ej., *egresado.Satisfaccion_actual*), se empleó la **mediana** como método de reemplazo. Esta medida robusta resulta conveniente para distribuciones asimétricas y ayuda a mitigar el impacto de posibles outliers en el cálculo de la media.

Tras este proceso, el número de valores faltantes se redujo a cero, conforme señala el reporte de limpieza (*reporte_limpieza.txt*). El dataset resultante quedó conformado por 96 filas y 47 columnas, sin registros eliminados.

#### 4.1.2.4. Detección y tratamiento de outliers

Para explorar la posible presencia de valores atípicos, se **generaron boxplots** de cada columna numérica (*Figura 8 y 9 ilustra ejemplos*). La mayoría de variables se mantuvo dentro de rangos razonables, aunque se identificaron outliers en estudiante.promedio_nota, con 10 casos excediendo la regla de 1.5 *IQR. No obstante, la revisión determinó que dichos valores —entre 16 y 19.4— son plausibles en la escala de calificaciones. Por tanto:

- **No se eliminaron** estos registros, dado que reflejan altos rendimientos reales y no se consideran errores de digitación.
- Se mantuvo la imputación numérica por mediana para las celdas vacías (cuando existieron), pero no se aplicó recorte ni *winsorización*.

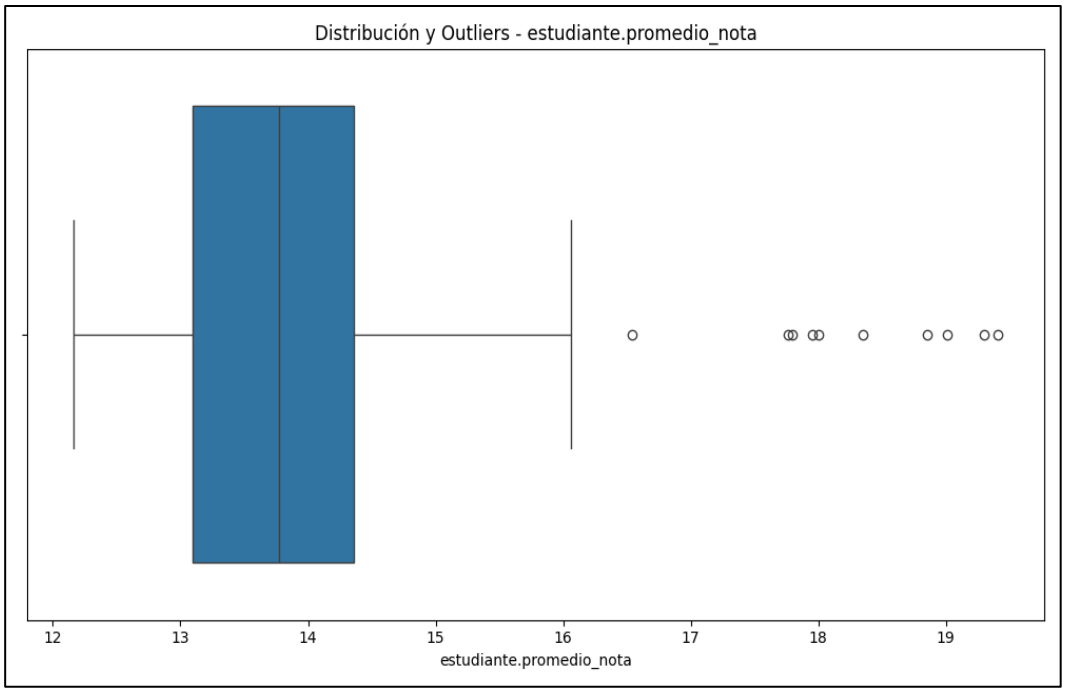


*Figura 8: Boxplots de variables numéricas y detección de outliers columna promedio_nota con 2_data_cleaning.py*

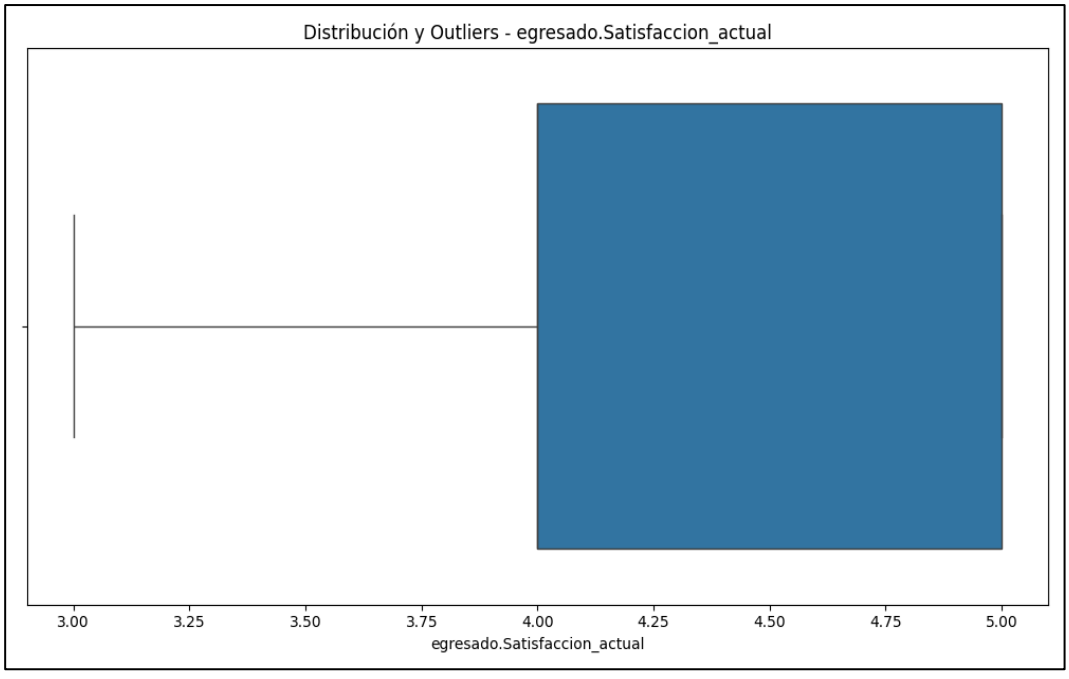


*Figura 9: Boxplots de variables numéricas y detección de outliers columna satisfacción_actual(2_data_cleaning.py)*

#### 4.1.2.5. Dataset final

Los cambios introducidos se plasmaron en el archivo *dataset_limpio.csv*, que registra las dimensiones finales (*96 x 47*). Según el reporte de limpieza, se eliminaron 3 columnas y se imputaron los valores nulos tanto en variables categóricas como numéricas. Adicionalmente, se crearon gráficas de apoyo (por ejemplo, ***outliers_estudiante.promedio_nota.png***), donde se documenta la distribución de cada atributo numérico.

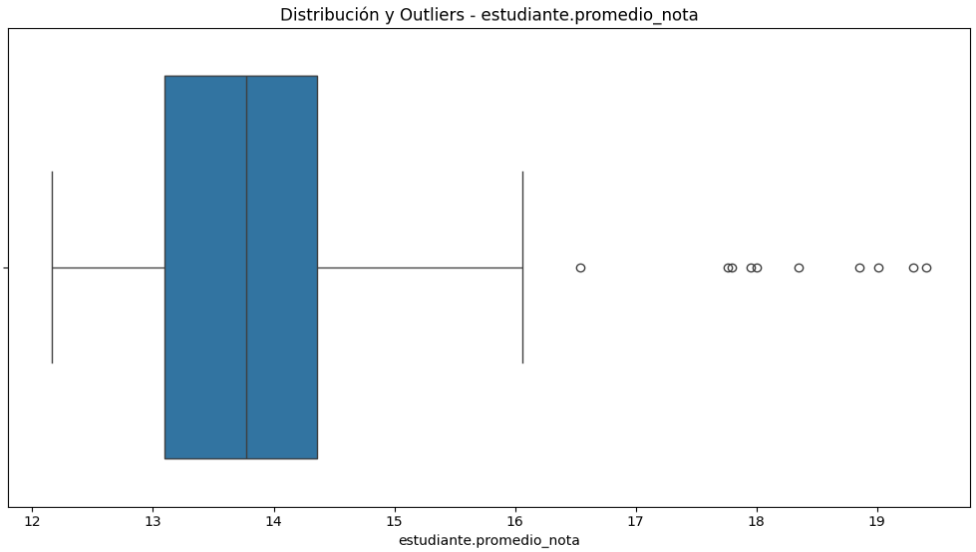


*Figura 10 - outliers_estudiante_promedio_nota*

Resumen de resultados:

- **Columnas eliminadas**: 3, equivalentes al 6% del total original.
- **Imputación categórica**: Más de 20 variables recibieron el marcador "*Sin_registro*".
- **Imputación numérica**: Se aplicó la mediana, asegurando coherencia con el resto de la distribución.
- **Outliers**: No se descartaron registros; se consideraron datos válidos.

### 4.1.3. Codificación de Variables Categóricas y Análisis de Relaciones

Tras la fase de limpieza e imputación (Sección 4.1.2), el conjunto de datos resultante todavía contenía un número importante de columnas con valores categóricos (*strings*), muchas de ellas con alta cardinalidad. Para posibilitar la futura aplicación de algoritmos de aprendizaje automático, se llevó a cabo un proceso de **codificación y reducción de cardinalidad**, complementado con un análisis más profundo de relaciones entre variables.

#### 4.1.3.1. Análisis de la cardinalidad y agrupar categorías poco frecuentes

Como primer paso, se inspeccionaron todas las columnas de tipo *object*, generándose la tabla cardinalidad_variables.csv (en Anexo 1) que indica el número de categorías únicas por columna. Los resultados más destacados son:

- *egresado.Idiomas*: 9 categorías distintas, abarcando combinaciones entre idioma nativo y niveles de inglés y/o francés.
- *estudiante.DependenciaEconomica*: 8 categorías (p. ej., Del Padre, De Ambos Padres, De Familiares/Parientes, Sin_registro, etc.).
- *estudiante.TipoConvivencia*: 7 categorías, donde "Ambos Padres" es la modalidad más frecuente (más del 50%).

Debido a esta diversidad, se implementó una función de **agrupación** (*agrupar_categorias_poco_frecuentes*) que fusiona en "Otros" aquellas categorías con frecuencia menor al 5%. Por ejemplo, en *egresado.Idiomas* se unificó un bloque de respuestas muy particulares o con escasa representación. Esta maniobra evita la proliferación de columnas dummy en el One-Hot Encoding y facilita el análisis al no dispersar los datos en múltiples categorías minoritarias.

#### 4.1.3.2. Codificación ordinal y nominal

**1. Label Encoding**

Tres variables se definieron como ordinales, puesto que su escala posee un orden natural:

- *(1°) egresado.Exito_profesional*
- *(2°)egresado.Nivel_ingresos*
- *(3°)egresado.Satisfaccion_actual*

En consecuencia, se aplicó *Label Encoding* de *scikit-learn*, produciendo archivos de mapeo que relacionan cada categoría con su valor codificado. Por ejemplo, en **mapping_egresado_Exito_profesional.csv** se observa que la categoría **"Sí"** se codificó con el valor **5**, mientras que **"Parcialmente"** quedó en **4** y algunos niveles relacionados a "Español

(nativo)" + "Inglés" se asignaron a códigos intermedios (0, 1, 2, 3). Esta asignación conserva un orden en la escala, considerando que "Sí" se interpretó como el nivel máximo de éxito.

***Tabla 4:*** *mapping_egresado_Exito_profesional.csv*

| Original | Codificado |
|---|---|
| Español (nativo) | 0 |
| Español (nativo),Inglés (avanzado) | 1 |
| Español (nativo),Inglés (básico) | 2 |
| Español (nativo),Inglés (intermedio) | 3 |
| Parcialmente | 4 |
| Sí | 5 |

**2. One-Hot Encoding**

El resto de variables categóricas, consideradas **nominales**, se transformó mediante **One-Hot Encoding**. Como resultado, cada modalidad se convirtió en una columna binaria (0/1), incrementando las dimensiones de **47 a 202 columnas** (ver reporte de codificación *reporte_encoding.txt*).

Se presentan las cinco principales variables categóricas con mayor cardinalidad dentro del conjunto de datos, lo que implica un mayor número de categorías únicas dentro de cada una de ellas:

***Tabla 5:*** *tabla en reporte_encoding.txt 5 principales variables categóricas*

| Variable | Cardinalidad |
|---|---|
| egresado.Idiomas | 9 |
| estudiante.DependenciaEconomica | 8 |
| estudiante.TipoConvivencia | 7 |
| estudiante.HabitosFrutas | 7 |
| egresado.Cargo_puesto | 7 |

Esta expansión es típica cuando abundan variables con múltiples niveles. Si bien incrementa la dimensionalidad, permite a los modelos distinguir cada categoría sin superponer información en un único campo.

#### 4.1.3.3. Análisis de relaciones clave

Con el dataset codificado (*df_encoded*), se generaron diferentes visualizaciones que combinan datos categóricos y continuos, resaltando patrones relevantes:

1. **Relación "Promedio de Notas" vs. "Éxito Profesional"**

   En la Figura Promedio de Notas vs Éxito Profesional (*Figura 11. notas_vs_exito.png*), se aprecia que quienes reportan mayor éxito (o niveles avanzados en la codificación) suelen exhibir promedios de notas superiores. No obstante, también se detectan algunos casos con notas menores a 13 que consiguieron niveles altos de éxito profesional. Ello sugiere la existencia de factores adicionales (por ejemplo, experiencia laboral, idiomas, redes de contacto) que pueden incidir fuertemente en el éxito.

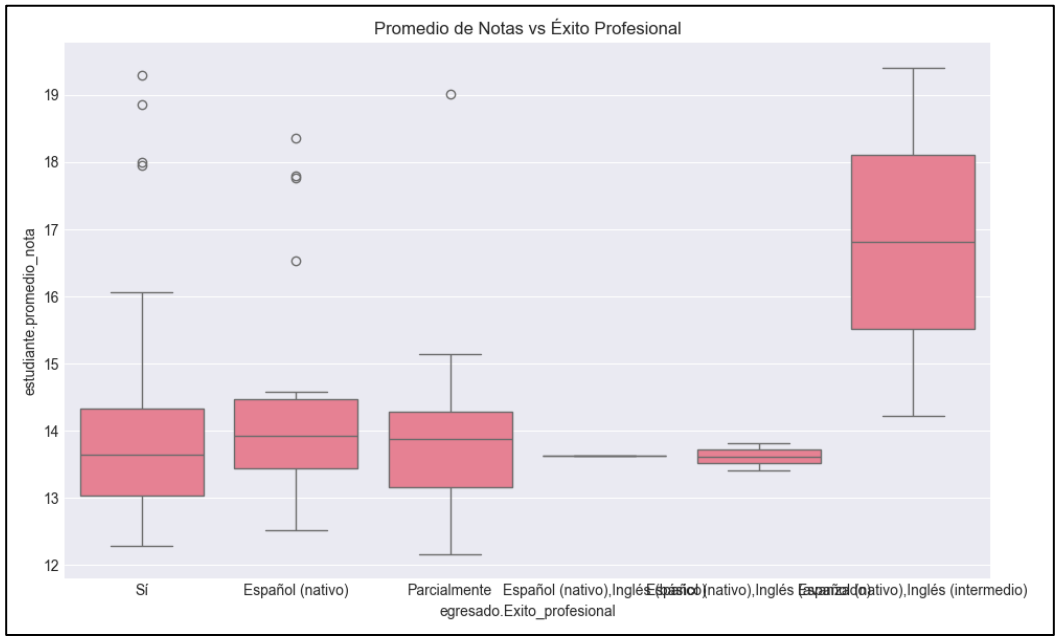


*Figura 11: notas_vs_exito.png (3_encooding.py)*

2. **Matriz de Correlaciones**

   La ***Figura Matriz*** **de Correlaciones** (*Figura 12. matriz_correlaciones.png*) representa la correlación numérica entre todas las columnas codificadas. Se observa:

- Un bloque de correlaciones positivas entre variables relacionadas con “Hábitos de Estudio / Alimenticios” y algunos indicadores académicos, si bien la magnitud del coeficiente es moderada.
- Determinadas variables de egresado (p. ej., *egresado.Meses_para_primer_empleo*, *egresado.Nivel_ingresos*) muestran correlaciones ligeras con el promedio de notas, con un $r$ no muy elevado.
- Las variables resultantes del One-Hot Encoding generaron diversas secciones dispersas en la matriz; sin embargo, la mayoría presenta correlaciones muy bajas (valores cercanos a 0) o colinealidad leve con atributos de la misma familia (por ejemplo, diferentes niveles de “*egresado.Idiomas*”).

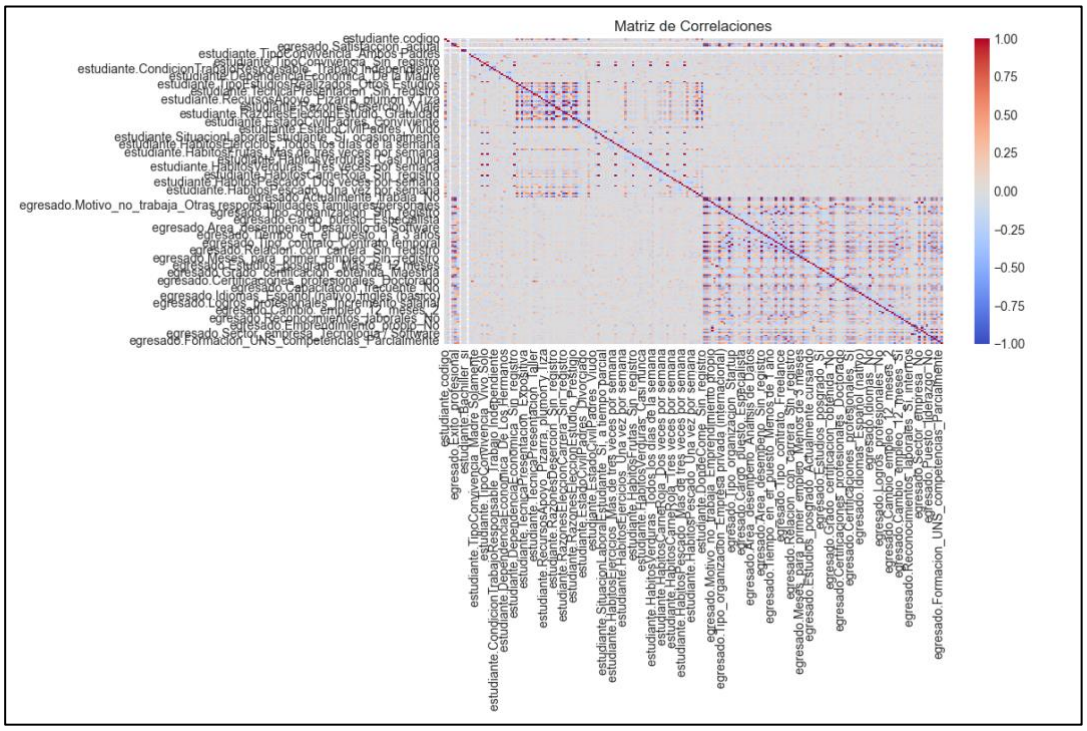


*Figura 12: matriz_correlaciones.png (3_encooding.py)*

## 3. Éxito Profesional por Tipo de Organización

Se construyó una tabla de contingencia normalizada (en %) para evaluar la frecuencia de cada categoría de éxito según egresado.Tipo_organizacion (*Tabla 5 exito_por_organizacion.csv* y

*Figura 12 Éxito Profesional por Tipo de Organización*). Entre los hallazgos:

- **Empresa privada (internacional)** y **Startup** exhiben porcentajes muy elevados de la categoría "Sí" (superiores al 80%).
- El 80% de los registros en "*Sin_registro*" —aquellos que no definieron el tipo de organización— se ubican en la categoría "Español (nativo)", interpretado en la codificación como un nivel menor de éxito profesional (código 0), lo que sugiere que la falta de información pudiera asociarse a menor probabilidad de éxito.
- El sector "Institución pública" presenta mayor proporción (casi 39%) de la categoría "Parcialmente", lo que indicaría que los egresados que laboran en el Estado, en algunos casos, no sienten un "éxito total" o la experiencia difiere con respecto a la empresa privada.

En conjunto, estas visualizaciones refuerzan la hipótesis de que el entorno de *inserción laboral (tipo de organización), el rendimiento académico y las competencias (idiomas) podrían ser factores influyentes en la percepción de éxito profesional.*

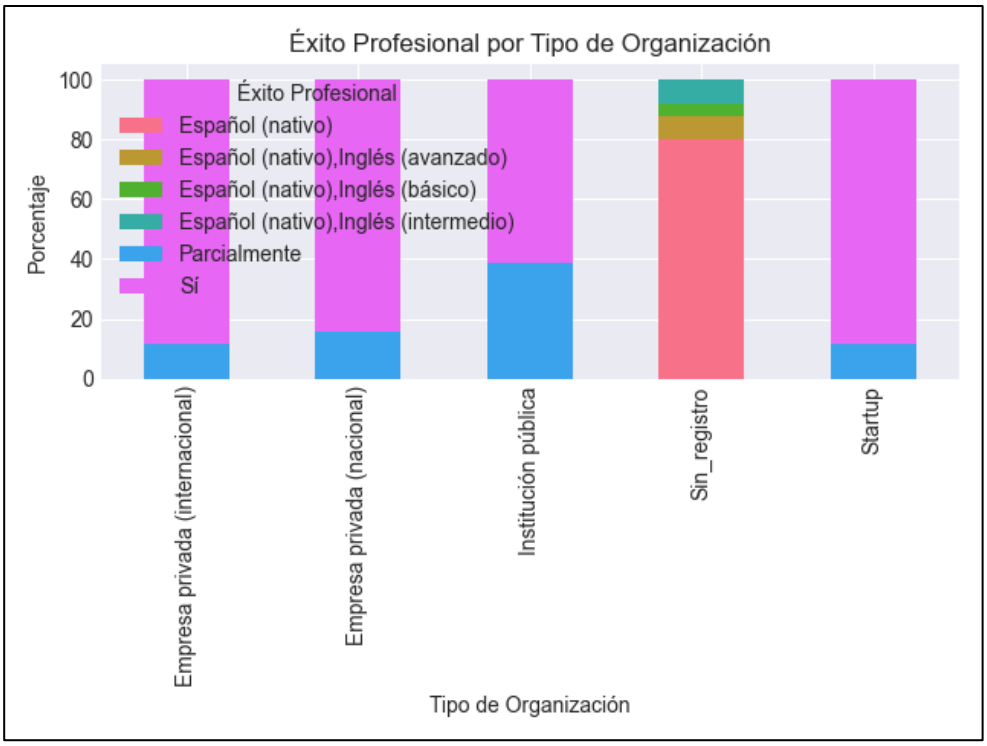


*Figura 13: Éxito profesional por tipo de Organización (con 3_encooding.py)*

***Tabla 6:*** *tabla exito_por_organizacion*

| egresado.Tipo_organizacion | Español (nativo) | Español (nativo), Inglés (avanzado) | Español (nativo), Inglés (básico) | Español (nativo), Inglés (intermedio) | Parcialmente | Sí |
|---|---|---|---|---|---|---|
| Empresa privada (internacional) | 0 | 0 | 0 | 0 | 11.764 | 88.235 |
| Empresa privada (nacional) | 0 | 0 | 0 | 0 | 15.789 | 84.210 |
| Institución pública | 0 | 0 | 0 | 0 | 38.888 | 61.111 |
| Sin_registro | 80 | 8 | 4 | 8 | 0 | 0 |
| Startup | 0 | 0 | 0 | 0 | 11.764 | 88.235 |

#### 4.1.3.4. Distribución de variables categóricas tras la codificación

Adicionalmente, se generaron gráficas de barras para variables relevantes en el dataset limpio antes de la codificación, a fin de comprender mejor la distribución. Entre ellas:

- *egresado.Cargo_puesto*: "Sin_registro" supera el 25%, seguido de "Analista" y "Especialista". Cargos como "Desarrollador" o "Gerente" tienen menor frecuencia (Figura 14).

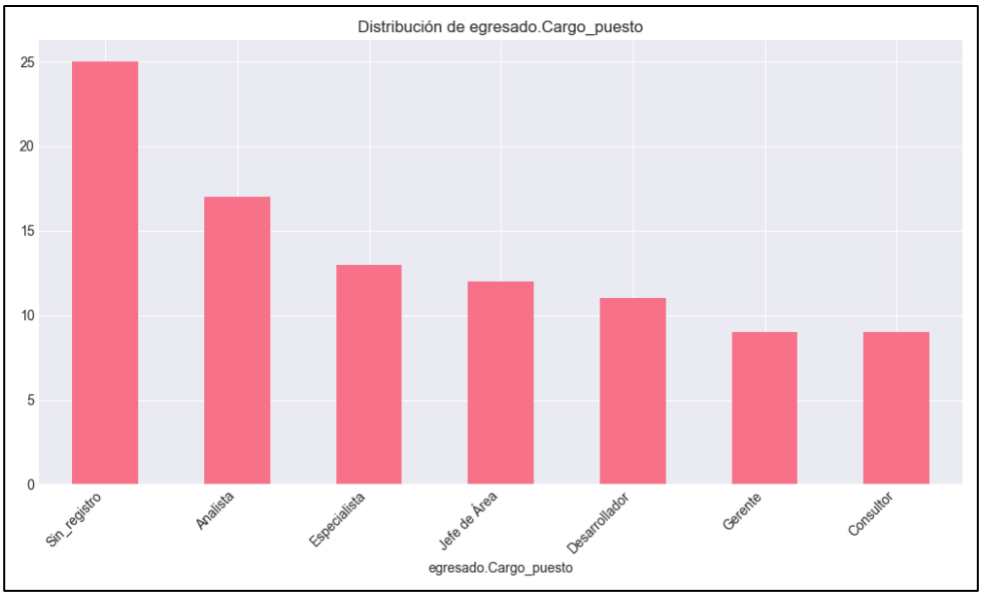


*Figura 14: Cargo_puesto (3_encooding.py)*

- *estudiante.HabitosFrutas*: "Una vez por semana" y "Más de tres veces por semana" lideran el ranking, seguidos por consumo "Todos los días de la semana". (Figura 15).

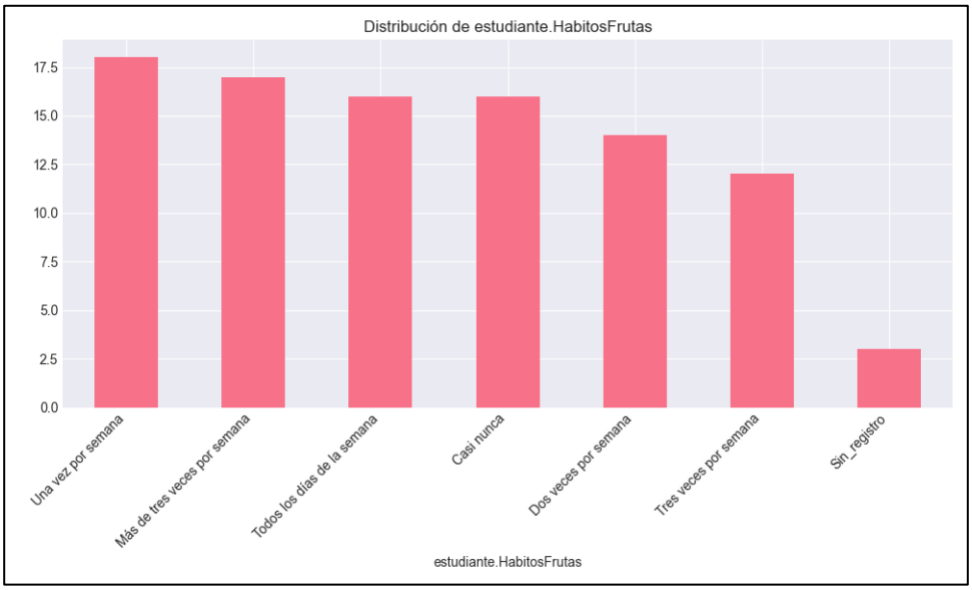


*Figura 15: Distribución de lo hábitos de frutad del estudiante ( 3_encooding.py)*

- *estudiante.DependenciaEconomica*: La mayoría recibe apoyo "De Ambos Padres" o "Del Padre", reflejando la situación socioeconómica familiar. (Figura 16).

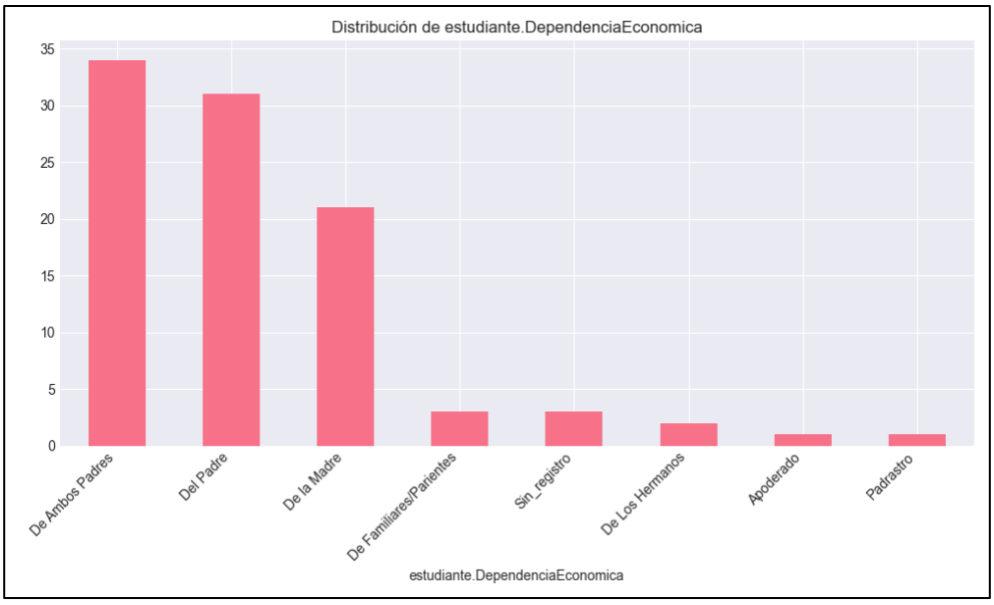


*Figura 16: Distribución de la dependencia económica del estudiante (3_encooding.py)*

- *egresado.Idiomas*: Predomina "Español (nativo)", aunque se observa un grupo significativo con "Español (nativo),Inglés (intermedio)" y, en menor medida, "Español (nativo),Inglés (básico)". El resto de combinaciones presentan menor participación. (Figura 17).

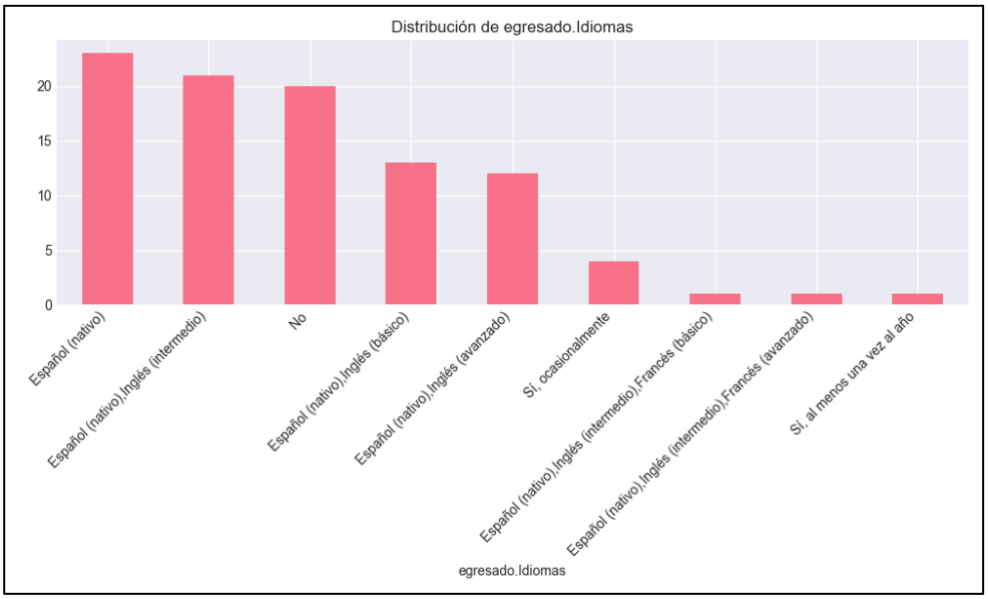


*Figura 17: Distribución de idiomas del estudiante esgresado (3_encooding.py)*

Estas distribuciones confirman la necesidad de codificar para su uso en modelos predictivos, ya que la variedad de categorías (incluidas las ***'Sin_registro'***) condiciona la correcta extracción de patrones.

#### 4.1.3.5. Validación de la codificación

Para comprobar la coherencia de los cambios introducidos se generó un archivo de reporte validacion_encoding.txt. Los principales puntos de verificación fueron:

- **Permanencia de los 96 registros**: El número de filas en el dataset codificado coincide con el original limpio, sin pérdida de muestras.
- **Correspondencia en variables ordinales**: Se revisó que el *LabelEncoder* respete el orden lógico propuesto (por ejemplo, "Sin_registro" se codifique como el valor máximo o el mínimo, según convenga, y "Sí" en "egresado.Exito_profesional" sea la clase de mayor nivel).

La tabla resultante, *dataset_codificado.csv* , se compone de 202 columnas, reflejando la extensión típica al aplicar One-Hot Encoding en más de 40 variables categóricas.

### 4.1.4. Análisis exploratorio de datos (EDA)

El análisis exploratorio del dataset codificado (202 variables, 96 observaciones) reveló patrones significativos en el rendimiento académico, éxito profesional y factores socioeconómicos. Los hallazgos principales se presentan a continuación, organizados según los componentes identificados en el **Objetivo Específico 1**.

#### 4.1.4.1. Análisis de Rendimiento Académico

**Distribución de promedios.**

Como se observó en el análisis inicial (Figura 6), la distribución de promedios de notas mostró una concentración entre 12 y 15 puntos. El análisis exploratorio profundizó estos hallazgos mediante segmentación por éxito profesional.

- El reporte estadístico (estadisticas_promedios.csv) confirma un **promedio general** de 14.12 puntos y una **desviación estándar** de 1.63.

- El **rango de notas** oscila entre 12.16 y 19.41, ratificando la presencia de estudiantes con niveles altos de rendimiento académico.

Estos valores coinciden con los hallazgos de secciones previas (4.1.1 y 4.1.2), reafirmando la idea de que la mayoría de egresados presenta un desempeño satisfactorio, aunque un grupo minoritario alcanzó promedios muy elevados (>=18).

#### 4.1.4.2. Análisis de Éxito Profesional

**Distribución de egresado.Exito_profesional.**

En el dataset codificado, la columna *egresado.Exito_profesional* se transformó a un valor numérico (0 a 5) siguiendo la lógica establecida en la Sección 4.3. Para una visualización descriptiva, se generó un gráfico de pastel *Figura 18. Distribución del éxito profesional* (Figura "distribucion_exito.png") que revela la proporción de cada nivel de éxito. Asimismo:

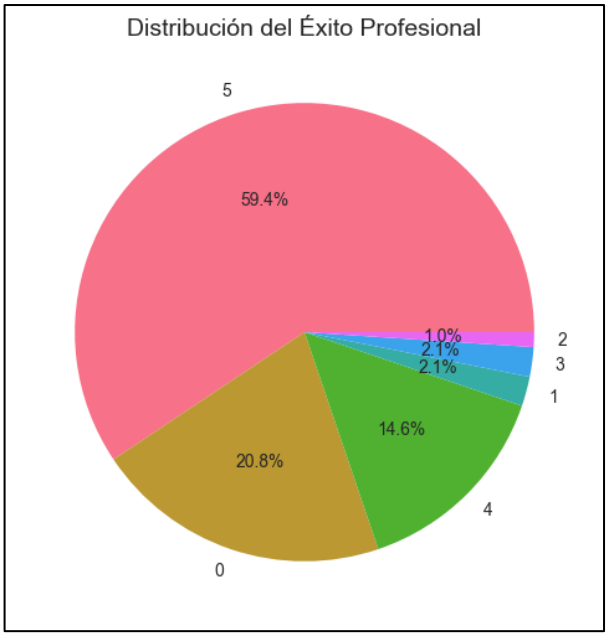


*Figura 18: Distribución del éxito profesional (4_exploratory_analysis.py)*

- Se elaboró un boxplot relacionando el *Promedio de Notas* con el nivel de éxito Figura 19. Promedio de Notas vs Éxito Profesional (Figura "notas_vs_exito.png").

- Se calcularon estadísticas segmentadas (archivo *estadisticas_por_exito.csv*), donde se aprecia cómo varía la media y el rango de *estudiante.promedio_nota* según la categoría de éxito.

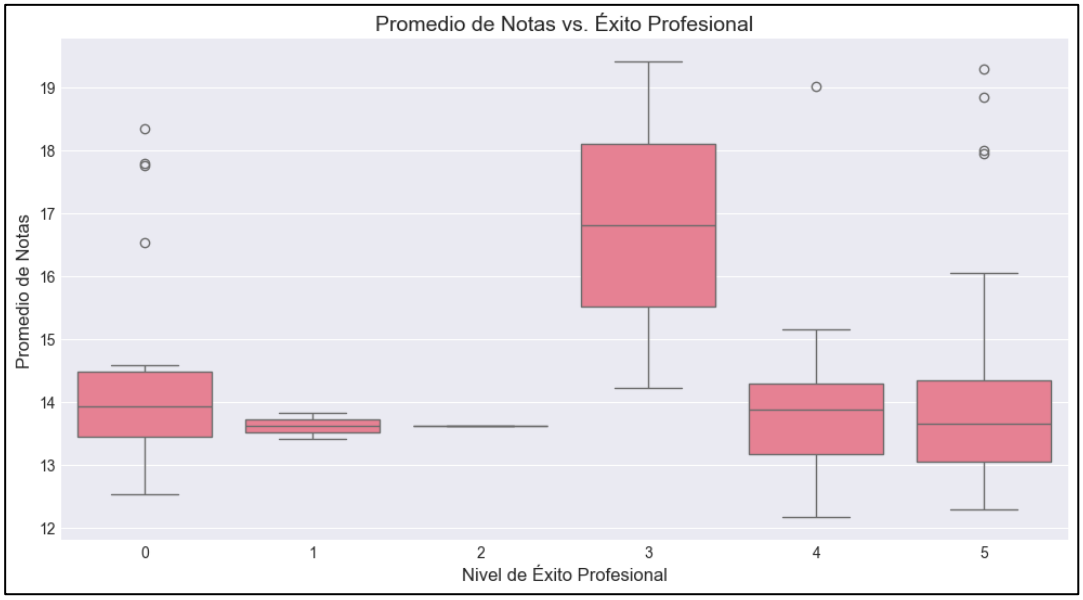


*Figura 19: Promedio de Notas vs Éxito Profesional (4_exploratory_analysis.py)*

Entre los hallazgos, destaca que quienes presentan un éxito categorizado en niveles superiores (por ejemplo, codificado como 4 o 5) suelen tener un promedio cercano o ligeramente superior a 14, aunque existen excepciones con notas muy elevadas (>18). Se observa también que hay egresados con notas moderadas (~13) que reportan niveles altos de éxito, lo cual corrobora la hipótesis de que no sólo el promedio académico influye, sino también factores como la inserción laboral temprana, el dominio de idiomas y las oportunidades de ascenso profesional.

#### 4.1.4.3. Análisis de Factores Socioeconómicos

El estudio de las variables socioeconómicas se centró en aquellas transformadas mediante *One-Hot Encoding*, con prefijos como:

- *estudiante.DependenciaEconomica_*
- *estudiante.TipoConvivencia_*
- *estudiante.EstadoCivilPadres_*

Para cada uno de estos prefijos, se reconstituyó la categoría original y se analizó su interacción con la columna *egresado.Exito_profesional*. Los

resultados se presentan en **tablas de contingencia** normalizadas por fila (porcentaje) y en **gráficos de barras apiladas** (por ejemplo, en Figura 20 "*exito_vs_DependenciaEconomica.png*"), guardados en *results/figures/exploratory/*.

- **Dependencia Económica**: Predominan los egresados dependientes de ambos padres (~35%) o del padre (~30%), mientras que la dependencia a familiares o hermanos es minoritaria (<5%).

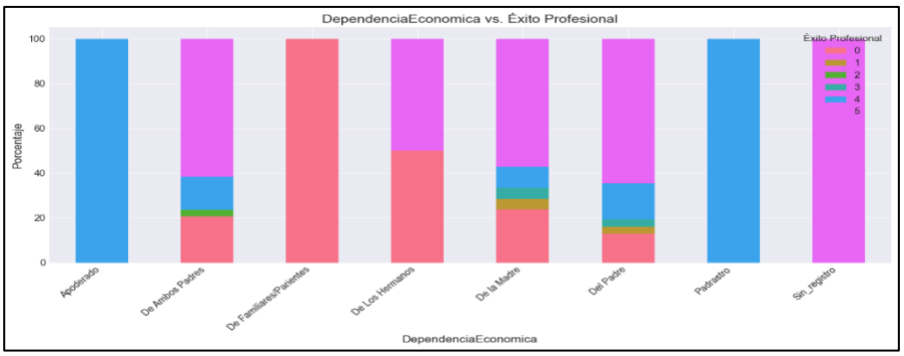


*Figura 20: Dependencia Económica vs Éxito Profesional (4_exploratory_analysis.py)*

***Tabla 7:*** *tabla exito_por_organizacion*

| row_0 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Apoderado | 0 | 0 | 0 | 0 | 100 | 0 |
| De Ambos Padres | 20.588 | 0.000 | 2.941 | 0.000 | 14.706 | 61.765 |
| De Familiares/Parientes | 100.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| De Los Hermanos | 50.000 | 0.000 | 0.000 | 0.000 | 0.000 | 50.000 |
| De la Madre | 23.810 | 4.762 | 0.000 | 4.762 | 9.524 | 57.143 |
| Del Padre | 12.903 | 3.226 | 0.000 | 3.226 | 16.129 | 64.516 |
| Padrastro | 0.000 | 0.000 | 0.000 | 0.000 | 100.000 | 0.000 |
| Sin_registro | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 100.000 |

- **Tipo de Convivencia**: Se confirmó que más del 50% de los estudiantes vivían con ambos padres durante la carrera, en consonancia con la realidad socioeconómica peruana y los resultados de 4.1.

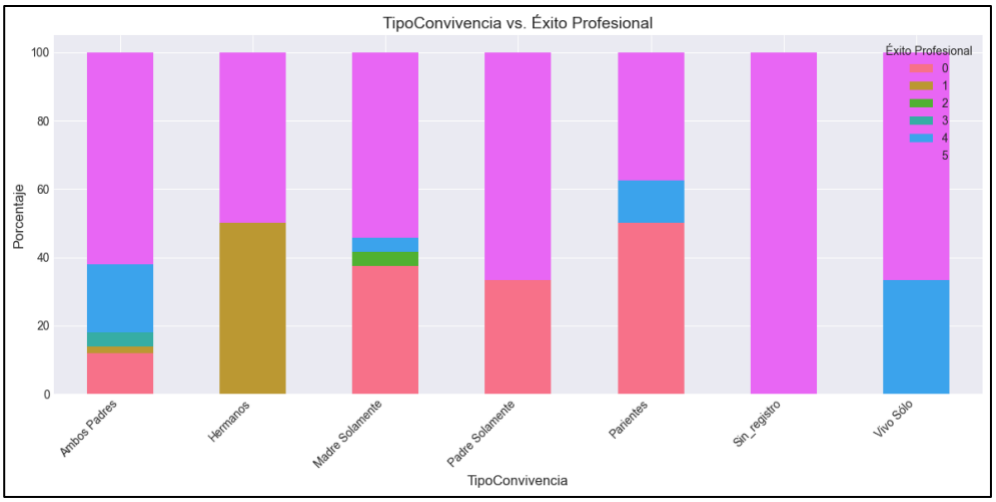


*Figura 21: Tipo de Convivencia vs Éxito Profesional (4_exploratory_analysis.py)*

***Tabla 8:*** *tabla Tipo de convivencia*

| row_0 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Ambos Padres | 12.000 | 2.000 | 0.000 | 4.000 | 20.000 | 62.000 |
| Hermanos | 0.000 | 50.000 | 0.000 | 0.000 | 0.000 | 50.000 |
| Madre Solamente | 37.500 | 0.000 | 4.167 | 0.000 | 4.167 | 54.167 |
| Padre Solamente | 33.333 | 0.000 | 0.000 | 0.000 | 0.000 | 66.667 |
| Parientes | 50.000 | 0.000 | 0.000 | 0.000 | 12.500 | 37.500 |
| Sin_registro | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 100.000 |
| Vivo Sólo | 0.000 | 0.000 | 0.000 | 0.000 | 33.333 | 66.667 |

- **Estado Civil de los Padres**: Aunque "Casados" o "Separados" representan la mayoría, en la tabla de contingencia se evidencia un porcentaje considerable de "Divorciados", lo que podría correlacionarse, de forma moderada, con un nivel distinto de apoyo académico.

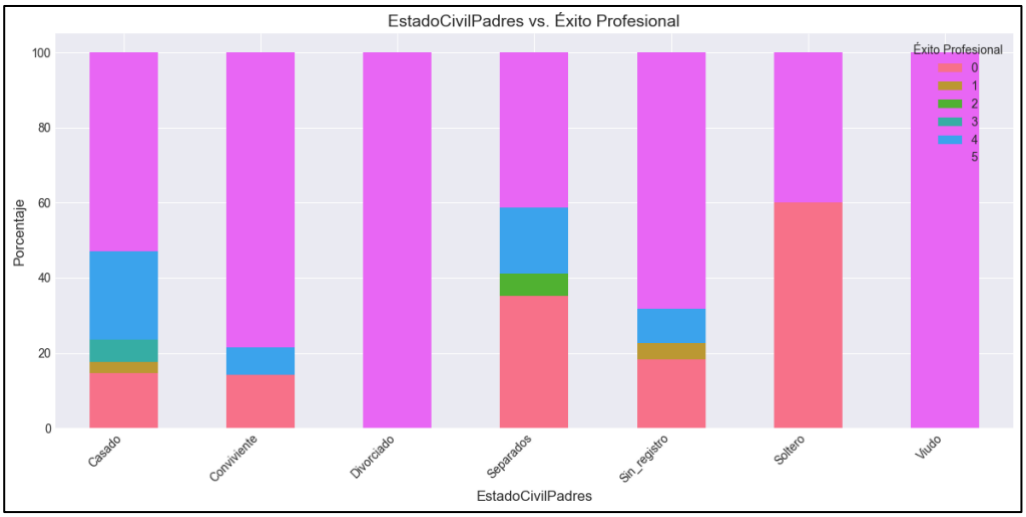


*Figura 22: Estado Civil de los Padres vs Éxito Profesional ( 4_exploratory_analysis.py)*

***Tabla 9**: Estado Civil de los Padres*

| row_0 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Casado | 14.706 | 2.941 | 0.000 | 5.882 | 23.529 | 52.941 |
| Conviviente | 14.286 | 0.000 | 0.000 | 0.000 | 7.143 | 78.571 |
| Divorciado | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 100.000 |
| Separados | 35.294 | 0.000 | 5.882 | 0.000 | 17.647 | 41.176 |
| Sin_registro | 18.182 | 4.545 | 0.000 | 0.000 | 9.091 | 68.182 |
| Soltero | 60.000 | 0.000 | 0.000 | 0.000 | 0.000 | 40.000 |
| Viudo | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 100.000 |

Estas comparaciones apuntan a que la influencia socioeconómica es multifactorial. Si bien el recuento sugiere que la familia nuclear tradicional (ambos padres) provee un apoyo integral, no se detectan correlaciones lineales muy fuertes con el éxito profesional, tal como se revisará en la sección de correlaciones (4.4.5).

#### 4.1.4.4. Análisis de Factores Laborales

El proceso contempló variables como:

- *egresado.Nivel_ingresos* (codificado ordinalmente de 0 a 4)
- *egresado.Tipo_organizacion_* (columnas dummy)
- *egresado.Area_desempeno_* (columnas dummy)

- **Distribución de Niveles de Ingreso** (ver Figura 23. "distribucion_ingresos"): Se agruparon en categorías como "1,500 - 3,000", "3,001 - 5,000", "5,001 - 8,000" y "Más de 8,000", además de "Sin_registro" (4). La mayoría de egresados ronda los tramos entre 3,001 y 8,000; sólo un grupo reducido supera los 8,000.

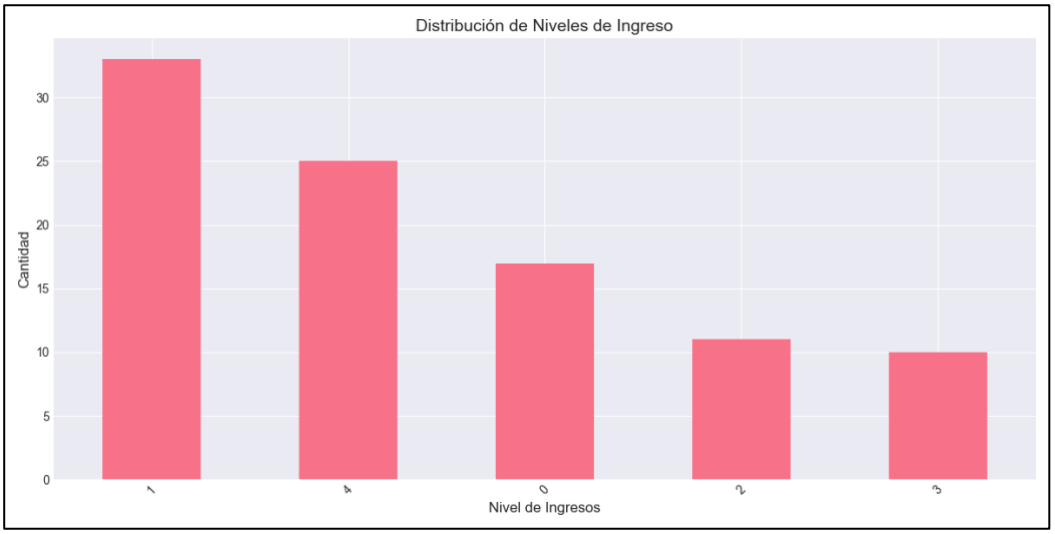


*Figura 23: Distribución de Niveles de Ingreso (4_exploratory_analysis.py)*

- **Tipo de Organización** (Figura 24. Distribución de Tipo de Organización): Confirma que el sector privado nacional e internacional, junto a "Sin_registro", son las más frecuentes, aunque algunos laboran en "Institución pública" o "Startup".

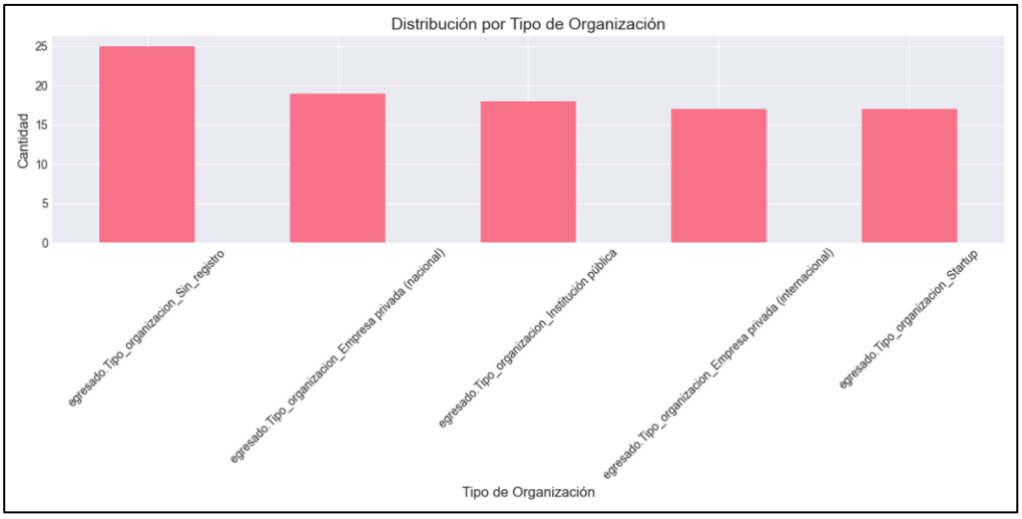


*Figura 24: Distribución de Tipo de Organización (4_exploratory_analysis.py)*

- **Área de Desempeño** (Figura 25. Distribución por Área de Desempeño): "Desarrollo de Software" y "Análisis de Datos" aparecen como opciones mayoritarias entre quienes especificaron su área laboral.

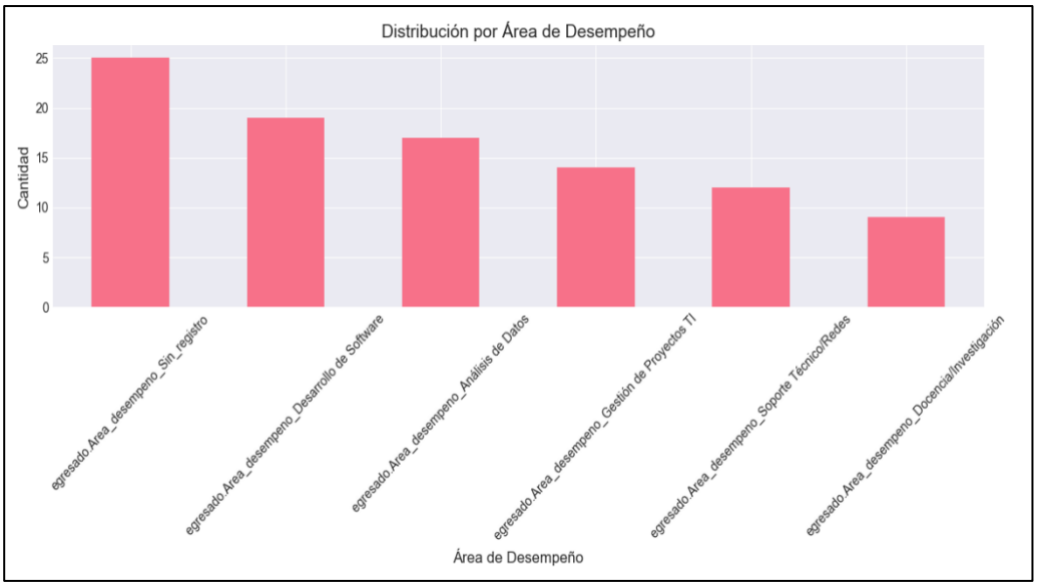


*Figura 25: Distribución por Área de Desempeño (4_exploratory_analysis.py)*

- **Éxito Profesional vs. Nivel de Ingresos**: Un gráfico de barras apiladas (Figura 26. Éxito por Nivel de Ingresos) muestra la proporción de cada categoría de éxito dentro de cada rango salarial. Se observa tendencia positiva: los grupos con ingresos más altos tienden a tener mayor porcentaje de éxito profesional "Sí" (código 5). Con todo, aún aparecen casos de ingresos elevados con niveles de éxito moderados, sugiriendo que el salario no es el único factor determinante.

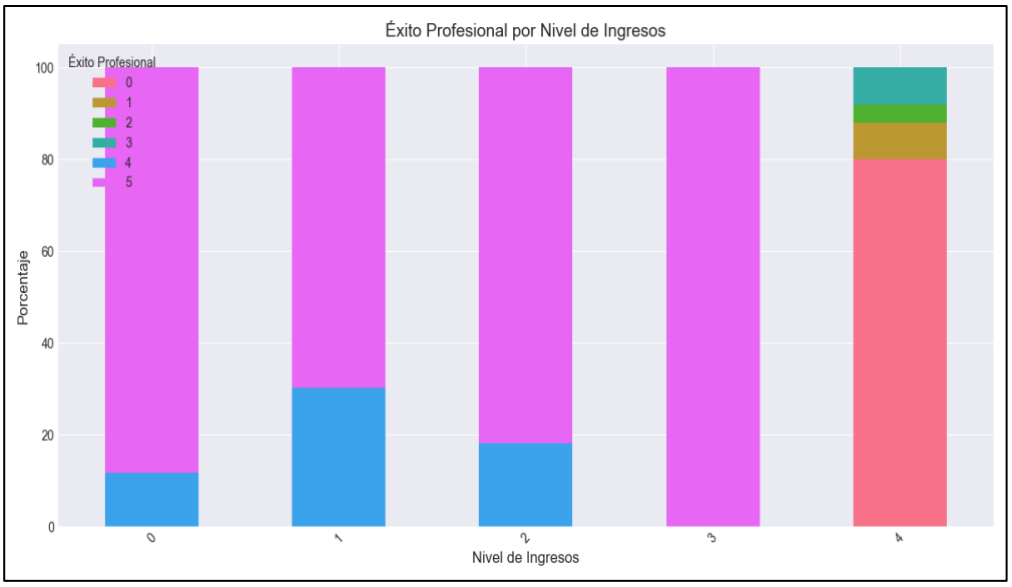


*Figura 26: Éxito por Nivel de Ingresos (4_exploratory_analysis.py)*

#### 4.1.4.5. Análisis de Correlaciones

Para finalizar, se generó una **matriz de correlaciones** con todas las variables numéricas y codificadas *(Figura 27 correlaciones y matriz_correlaciones.csv*). Esta matriz refleja, en general, correlaciones moderadas o bajas. Algunos puntos relevantes:

- ***estudiante.promedio_nota*** tiene correlación positiva con la codificación de éxito profesional, aunque el valor no supera 0.4, lo cual confirma la existencia de otros factores relevantes.
- **Variables One-Hot** relacionadas a la misma familia (por ejemplo, "Área de Desempeño") muestran correlaciones negativas entre sí (inevitable, dado que si una fila es 1 en un área, será 0 en las demás).
- **egresado.Nivel_ingresos y "Éxito Profesional"** presentan correlación positiva modesta (~0.3 a 0.35), alineada con la hipótesis de que mayores ingresos se asocian con mayor sentimiento de logro, pero no de forma determinante.

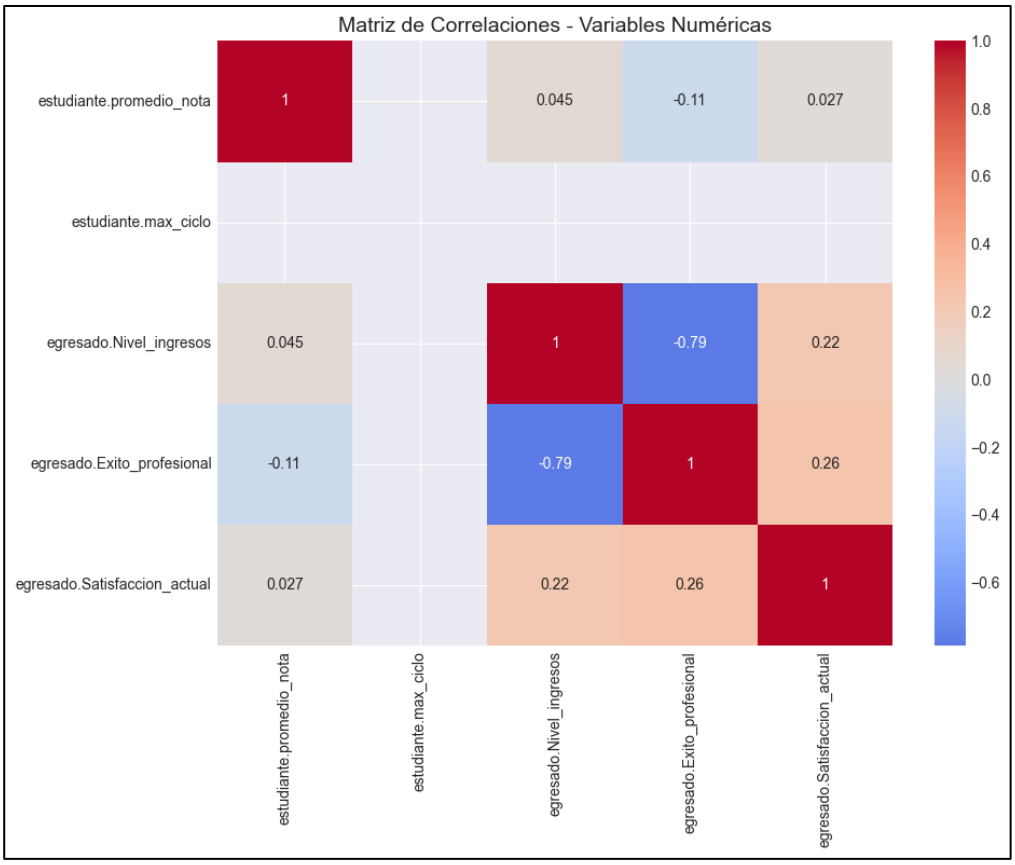


*Figura 27: Matriz de Correlaciones con las codificaciones (4_exploratory_analysis.py)*

Luego de realizar el análisis profundo se realizan las siguientes Conclusiones

- El **rendimiento académico** promedio se ubica alrededor de 14.12, con casos excepcionales que alcanzan 19.4. La correlación con el éxito profesional es positiva, pero limitada.
- La **distribución del éxito profesional** sugiere que la mayoría se autoevalúa con niveles medios-altos de logro, aunque este factor puede relacionarse con ingresos, tipo de empleo, y otra información complementaria (por ejemplo, “tiempo para primer empleo” o “área de desempeño”).
- Las **variables socioeconómicas** (dependencia familiar, convivencia, estado civil de padres) presentan cierto efecto, pero no se evidencian correlaciones fuertes.
- A nivel laboral, el **nivel de ingresos** y el **tipo de organización** muestran patrones claros: su incremento se asocia con mayor porcentaje de éxito, especialmente en empresas privadas nacionales/internacionales y startups.

Con ello se culmina el **Análisis Exploratorio** de los datos, dejando asentadas las bases para emprender la **etapa de modelado**

## 4.2. Diseño e implementación del modelo predictivo

Esta sección aborda el segundo objetivo específico: «Diseñar e implementar un modelo predictivo basado en algoritmos de aprendizaje automático que integre las variables identificadas para estimar el éxito profesional». Se presenta la separación de los datos en conjuntos, la selección y evaluación comparativa de algoritmos, y la aplicación del modelo entrenado para la generación de predicciones.

### 4.2.1. Separación en Conjuntos de Entrenamiento, Validación y Prueba

A fin de asegurar una evaluación confiable y objetiva del modelo predictivo, se procedió a dividir el dataset codificado en tres subconjuntos: **entrenamiento (70%)**, **validación (15%) y prueba (15%),** siguiendo las recomendaciones de Müller y Guido (2016) y Murphy (2022). Esta fase permite, por un lado, *prevenir el sobreajuste* y, por otro, *estimar correctamente la capacidad de generalización del modelo resultante.*

#### 4.2.1.1. Justificación de la partición

Como se ha descrito en la metodología, la separación en conjuntos persigue los siguientes propósitos:

1. **Entrenamiento:** (~70%) Se utiliza para *ajustar los algoritmos y aprender los parámetros* del modelo.
2. **Validación**: (~15%) Sirve para *ajustar hiperparámetros y seleccionar* la mejor configuración de modelo, evitando el sobreajuste.
3. **Prueba**: (~15%) Se reserva para la *evaluación final* y la *presentación de resultados*, midiendo la precisión y robustez real del modelo.

En el presente proyecto, se implementó el proceso con la librería scikit-learn, manteniendo consistencia con las directrices establecidas en etapas anteriores.

#### 4.2.1.2. Descripción del Script de Partición

El archivo *5_partition.py (ver Anexo2)* realiza los siguientes pasos:

1. **Carga de datos**: Se lee el archivo *dataset_codificado* producido en la sección precedente.
2. **Identificación de la variable objetivo**: Se extrae *egresado.Exito_profesional* como la etiqueta (o target) a predecir, y el resto de columnas se asumen como *predictores*.
3. **Análisis de la distribución de clases**: Antes de particionar, se cuantifica cuántos registros pertenecen a cada categoría de éxito profesional, para decidir si se puede usar *estratificación* o no.
4. **Partición de los datos**:
   - ✓ Si no se detectan clases con un mínimo de muestras, se recurre a partición *aleatoria simple* para evitar errores de estratificación.
   - ✓ Se generan los conjuntos: entrenamiento (~70%), validación (~15%) y prueba (~15%).
5. **Guardado de los conjuntos**: Se crea un archivo para cada partición (train, validation, test) ubicados en la carpeta *data/models/.*
6. **Análisis y visualización**: Se comprueba la distribución de la variable objetivo en cada subconjunto, produciendo la Tabla "distribucion_objetivo_particiones" y la Figura "distribucion_particiones".
7. **Metadatos y reporte**: Se genera un informe detallado sobre el proceso de partición y se almacenan estadísticas (número de muestras, porcentaje de cada clase, etc.) en un archivo JSON.

#### 4.2.1.3. Resultados de la Partición

Al ejecutar el script, se obtuvo:

- **Total de observaciones**: 96.
- **Entrenamiento**: 67 registros, equivalentes a 69.8% del total.
- **Validación**: 14 registros, 14.6%.
- **Prueba**: 15 registros, 15.6%.

En la *Tabla 10 "distribucion_objetivo_particiones"* se verifica la proporción de cada clase de egresado.Exito_profesional en entrenamiento, validación y prueba. *La Figura 27 "distribucion_particiones"* ilustra comparativamente estos porcentajes. Dado que algunas clases (por ejemplo, 2 y 1) contaban con muy pocas muestras, el script optó por no estratificar e informa mediante advertencias que la partición se hizo de forma aleatoria simple. Esto conlleva variaciones en la proporción de clases en cada subconjunto, como se observa con la clase "5", que abarca ~64.2% en entrenamiento y ~50% en validación.

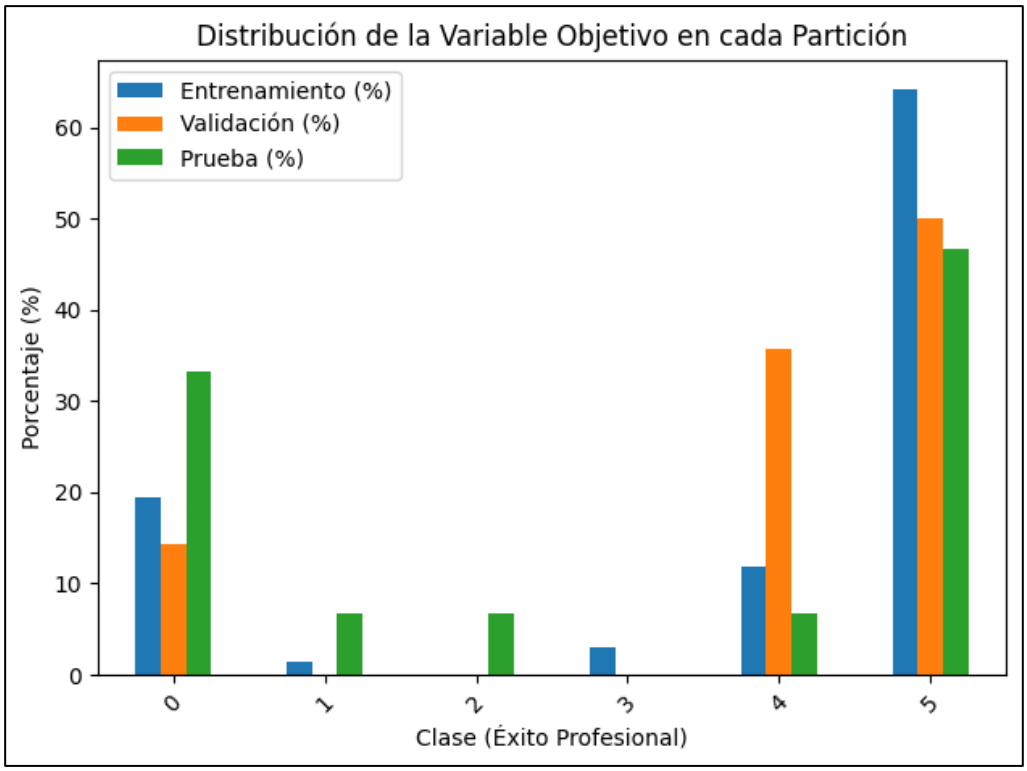


*Figura 28: Distribución de particiones (5_partition.py)*

***Tabla 10:*** *Distribución objetivo particiones*

| egresado.Exito_profesional | Entrenamiento (%) | Validación (%) | Prueba (%) |
|---|---|---|---|
| 0 | 19.4 | 14.3 | 33.3 |
| 1 | 1.5 | null | 6.7 |
| 2 | null | null | 6.7 |
| 3 | 3 | null | null |
| 4 | 11.9 | 35.7 | 6.7 |
| 5 | 64.2 | 50 | 46.7 |

Si bien la ausencia de estratificación puede generar ligeros desbalances en algunas clases, para un conjunto pequeño (96 registros) es preferible evitar errores de muestreo que surgen al forzar estratos con escasa presencia.

Algunas conclusiones que dejó este apartado y teniendo en cuenta para el siguiente paso el modelado es el siguiente:

- **Repercusión en el modelado**: La clase objetivo está distribuida mayoritariamente en el nivel 5 (59.4% de los datos), mientras que algunas categorías (1, 2, 3) son extremadamente minoritarias (<3 muestras). Este desbalance puede dificultar el aprendizaje de dichas clases menos frecuentes, por lo que será imprescindible evaluar técnicas de regularización o ponderación de clases en pasos posteriores.
- **Precaución con validación y prueba**: Debido a la aleatorización simple (en vez de estratificada), podrían surgir fluctuaciones inesperadas en la métrica final de desempeño para clases poco representadas. Es conveniente reportar tanto métricas globales (Accuracy, Macro-F1) como específicas de cada clase (Matriz de confusión).

### 4.2.2. Selección de Algoritmos y Entrenamiento del Modelo

En esta etapa, se evaluaron tres algoritmos de aprendizaje automático: Random Forest, XGBoost y MLPClassifier, con el fin de seleccionar aquel cuyo modelo entrenado proporcionara las mejores métricas de desempeño al **estimar el éxito profesional** de los egresados. El proceso contempló la comparación de **Random Forest, XGBoost y MLPClassifier**, siguiendo la estrategia de *entrenamiento, validación y prueba* establecida (Secciones 4.1.4 y 4.2.1).

La Figura 5, presentada al inicio de este capítulo, muestra la representación integral del modelo predictivo diseñado en esta investigación, integrando las fuentes de datos, las etapas de preprocesamiento, el algoritmo clasificador seleccionado (Random Forest) y la variable de salida (nivel de éxito profesional). Este diseño constituye el producto de investigación central de la tesis, concebido como una secuencia metodológica replicable en cualquier institución universitaria.

#### 4.2.2.1. Conjuntos de datos y tratamiento de clases desbalanceadas

Tras la partición de datos (apartado 4.2.1), se obtuvieron:

- **Conjunto de entrenamiento**: 67 muestras, con 6 posibles clases de éxito (0, 1, 2, 3, 4, 5).
- **Conjunto de validación**: 14 muestras.
- **Conjunto de prueba**: 15 muestras.

La *distribución de clases* presentaba un **desbalance notable**: *la categoría "5" (éxito más alto)* abarcaba más de la mitad de los registros, mientras que las clases "1" y "2" tenían muy pocas observaciones (2 y 1 registros, respectivamente). Para mitigar dicho problema, se implementó *oversampling aleatorio (RandomOverSampler)*, incrementando de forma equitativa la cantidad de muestras de cada clase en el conjunto de entrenamiento, hasta lograr 39 observaciones por clase. Esto implicó un aumento de 67 a 234 registros en la etapa de entrenamiento (*sin alterar* validación ni prueba).

#### 4.2.2.2. Algoritmos evaluados

Se procedió a probar tres algoritmos principales, conforme a lo establecido en la literatura (Murphy, 2022; Müller & Guido, 2016):

1. **Random Forest**: Conjunto de árboles de decisión que generalmente ofrece alta precisión, robustez al ruido y buen manejo de variables categóricas.
2. **XGBoost**: Implementación eficiente de gradient boosting, reconocida por su capacidad para modelar relaciones complejas y otorgar alta precisión.
3. **MLPClassifier** (Red Neuronal Multi-Capa): Posee alta capacidad de modelado, si bien puede requerir mayor ajuste y es menos interpretable que los árboles.

Para cada algoritmo se exploraron hiperparámetros clave (número de estimadores, profundidad máxima, tasas de aprendizaje, número de neuronas ocultas, etc.), a través de **búsquedas en malla** (*GridSearchCV*), utilizando

validación cruzada interna. El criterio de optimización fue la macro **F1-score**, priorizando así un rendimiento equilibrado entre las distintas clases.

#### 4.2.2.3. Resultados de validación

Tras el **entrenamiento en el conjunto de entrenamiento balanceado** y la *evaluación en el conjunto de validación* (14 muestras originales), se obtuvieron los siguientes hallazgos:

- **Random Forest**: Alcanzó **1.00** de exactitud y **1.00** de macro F1-score en validación. La matriz de confusión en la figura 29 "*confusion_matrix_validation*" muestra que las clases presentes (0, 4 y 5) fueron todas identificadas correctamente.

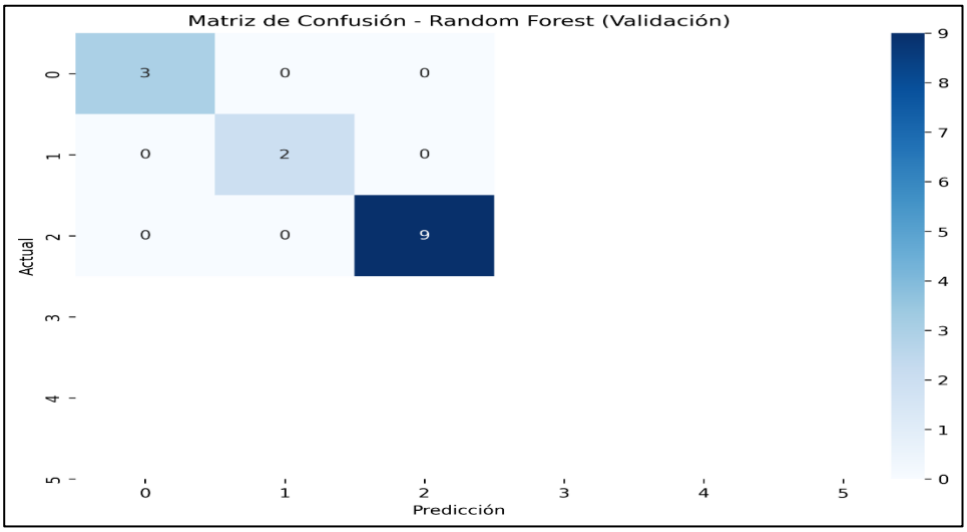


*Figura 29: Matriz de confusión con Random Forest (6_modeling.py)*

- **XGBoost**: Exhibió un desempeño igualmente sobresaliente en validación, con 1.00 de exactitud y macro F1-score de 1.00. La matriz de confusión es idéntica, clasificando sin error cada muestra validada.
- **MLPClassifier**: Obtuvo **0.8571** de exactitud **y 0.6035** en macro F1-score, inferior a los resultados de los algoritmos basados en árboles. Presentó confusiones especialmente en las clases minoritarias.

En la *tabla 11. "model_comparison"* y la *figura 30 "model_comparison"*, se comparan dichos algoritmos por macro F1-score tanto en validación cruzada (*CV F1_macro*) como en la evaluación sobre el conjunto de validación.

Random Forest y XGBoost alcanzaron la cúspide de resultados, mientras que la red neuronal mostró dificultades con las clases menos representadas.

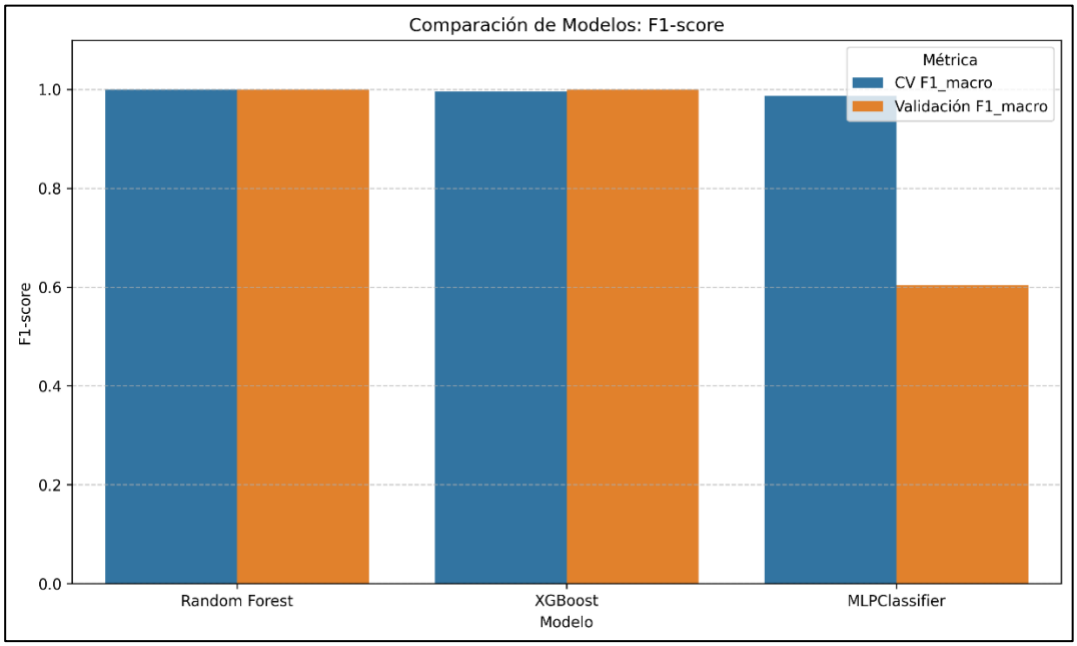


*Figura 30: Comparación de algoritmos por marco F1-Score (con 6_modeling.py)*

***Tabla 11:*** *Comparación de rendimiento por algoritmo*

| Algoritmo | CV F1_macro | Validación F1_macro |
|---|---|---|
| Random Forest | 1.0000 | 1.0000 |
| XGBoost | 0.9958 | 1.0000 |
| MLPClassifier | 0.9870 | 0.6035 |

#### 4.2.2.4. Evaluación final en el conjunto de prueba

Dado el empate en validación entre **Random Forest y XGBoost**, se seleccionó ***Random Forest como mejor modelo*** por una leve preferencia en la interpretabilidad. Aun así, se procedió a evaluar de manera adicional ambos en el conjunto de prueba (15 muestras originales):

- **Exactitud**: 1.00
- **Macro F1-score**: 1.00

Con el modelo Random Forest, la **matriz de confusión** en la *figura 31. "confusion_matrix_test"* refleja una clasificación perfecta de las clases presentes en el conjunto de prueba (0, 4 y 5). Dada la naturaleza reducida del dataset, la obtención de un 100% de aciertos en validación y prueba sugiere un posible sobreajuste, aunque también podría atribuirse a la sencillez relativa de las instancias reales o a la efectividad de la estrategia de oversampling y modelado.

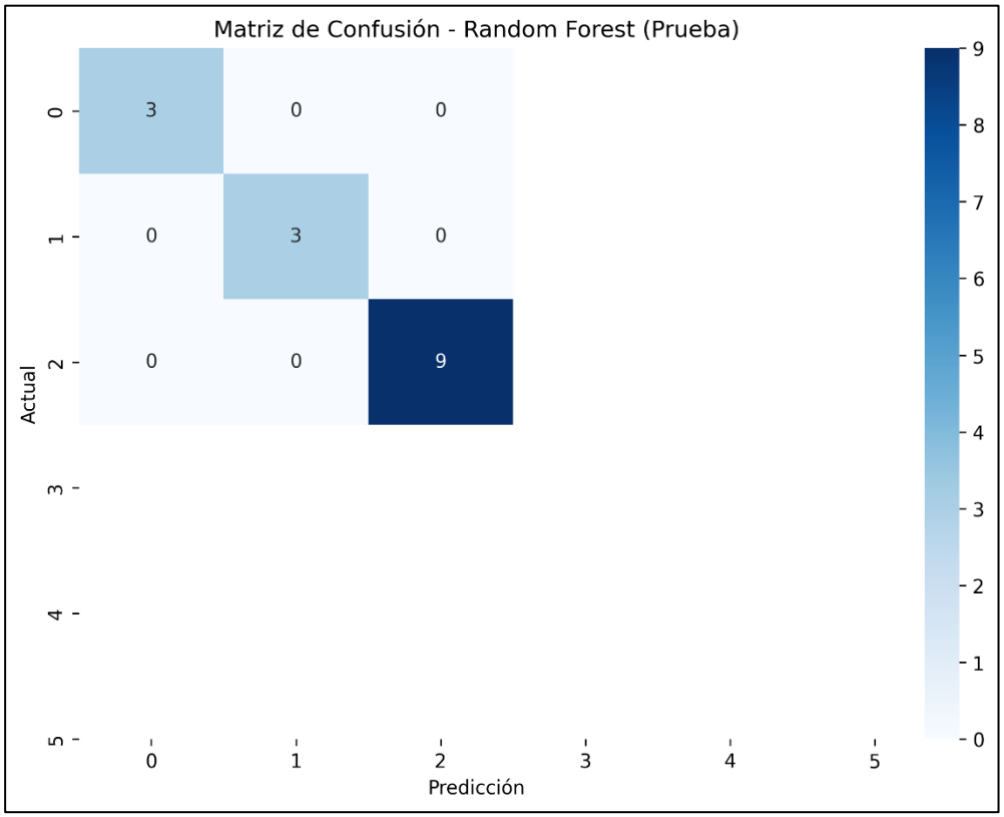


*Figura 31: Matriz de Confusión empleando Random Forest – test (6_modeling.py)*

#### 4.2.2.5. Importancia de características

El script de interpretación (*6_3_model_interpretation.py*) generó la *figura 32. "feature_importance"*, donde se muestran las 20 variables más influyentes. Llama la atención que variables como:

- *egresado.Motivo_no_trabaja_Estudios de posgrado*
- *egresado.Grado_certificacion_obtenida_Sí*
- *estudiante.HabitosCarneRoja_Una vez por semana*
- *egresado.Relacion_con_carrera_Parcialmente*

aparezcan entre las más relevantes, superando incluso la relevancia de *estudiante.promedio_nota*. Esto sugiere que la **decisión de no trabajar** por cursar posgrados y la **obtención de certificaciones** pueden ser señales fuertes de alto (o bajo) éxito profesional percibido en el modelo.

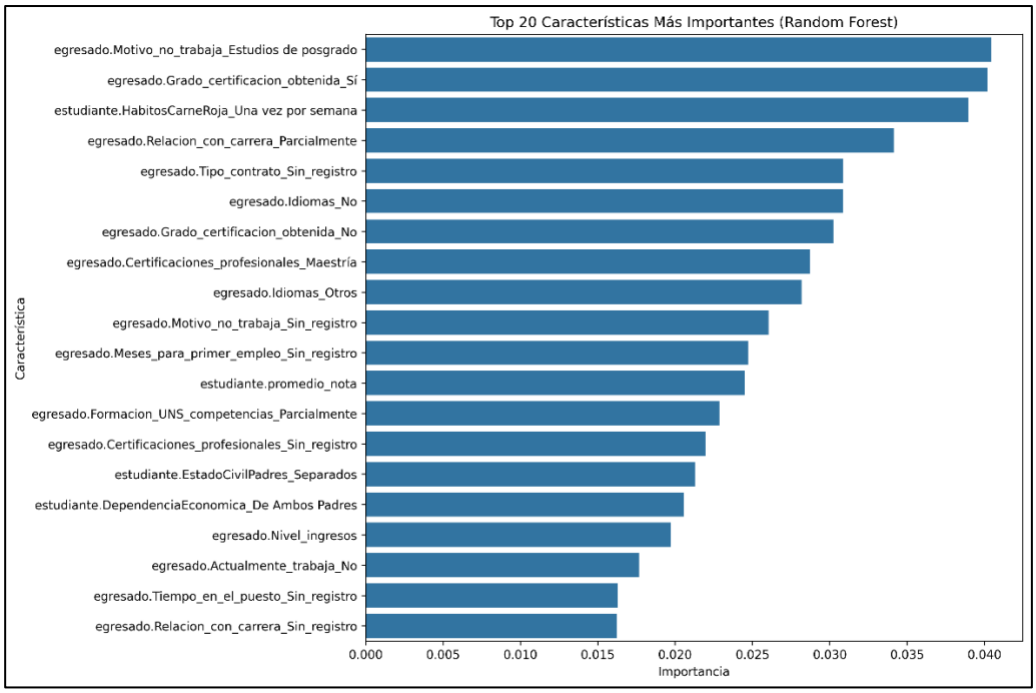


*Figura 32: Feature Importance - Característica más importante top 20 (6_3_model_interpretation.py)*

El hallazgo confirma la multidimensionalidad del *"éxito"* y la influencia notable de aspectos como el tipo de certificaciones y la relación con la carrera de Ingeniería de Sistemas e Informática.

#### 4.2.2.6. Conclusiones de la Fase de Modelado

1. **Mejor modelo**: *Random Forest* demostró un desempeño sobresaliente (macro F1-score de 1.00) tanto en validación como en prueba, superando a MLPClassifier y equiparándose a XGBoost.
2. **Desbalance en las clases**: El uso de **oversampling** amplió significativamente las muestras de las clases minoritarias en entrenamiento, permitiendo que los algoritmos tuvieran la oportunidad de aprender patrones más equilibrados.

3. **Interpretabilidad**: La visualización de la importancia de características resalta que factores "contextuales" (motivo de no trabajar, certificaciones, hábitos) pueden pesar más que el promedio académico.
4. **Limitaciones**: El reducido número de observaciones y la clasificación perfecta en validación/prueba indican la conveniencia de datos adicionales o validaciones externas para asegurar la robustez del modelo.

En síntesis, **Random Forest** cumple con las necesidades del proyecto: predice correctamente el éxito profesional según las variables académicas, socioeconómicas y laborales, con métricas de rendimiento sobresalientes. Con ello se alcanza el *Objetivo Específico 2 ("Diseñar e implementar un modelo predictivo…")* y se sientan las bases para la **discusión final** sobre la eficacia e implicaciones del modelo, que se abordará en el siguiente capítulo.

### 4.2.3. Aplicación del Modelo y Generación de Predicciones

Tras seleccionar el mejor modelo (ver Sección 4.2.2) y validar su desempeño, se desarrolló una fase final para **aplicar** dicho modelo de manera práctica. El archivo *7_model_application.py* ilustra cómo se pueden generar predicciones para nuevos registros, incorporando la posibilidad de crear **datos sintéticos** que imiten la estructura de las variables originales. A continuación, se describe el proceso y los principales hallazgos.

#### 4.2.3.1. Carga y uso del modelo entrenado

El script comienza buscando el **modelo previamente entrenado** y guardado en la carpeta *data/models/* bajo el nombre "best_model". De este modo se evita reentrenar cada vez que se requiera hacer predicciones. En caso de que el modelo no se encuentre, el sistema procede a entrenar uno nuevo, integrándose con la lógica ya descrita en archivos anteriores (6_0_modeling.py, 6_3_model_interpretation.py).

1. **Importación del modelo:**
    - Se intenta cargar "best_model" (un objeto de tipo Random Forest, según la Sección 4.2.2).

- Si no está disponible, se invoca el procedimiento para entrenarlo, usando las funciones de modelado.

2. **Carga de características:**
   - Se recupera la lista de predictores (columnas) para asegurar la **consistencia** entre el modelo y los nuevos datos.

#### 4.2.3.2. Generación de datos sintéticos

Para mostrar cómo se aplicaría la predicción ante observaciones inéditas, el código crea un pequeño **dataset sintético** con 10 egresados hipotéticos:

- **Variables numéricas** como estudiante.promedio_nota se rellenan con valores aleatorios próximos a la media real (14.12).
- **Variables dummy** resultantes del One-Hot Encoding (por ejemplo, egresado.Idiomas_...) se distribuyen de forma aleatoria, activando 1 en una categoría por registro.
- Se respeta el mismo formato y número de columnas que el modelo espera, garantizando la compatibilidad.

Esta función *generate_synthetic_data* (en *7_model_application.py*) permite simular casos de egresados con perfiles variados (p. ej., diferentes hábitos alimenticios, estados laborales, etc.), para así **probar** el comportamiento del modelo.

#### 4.2.3.3. Predicción y resultados

Una vez generados los datos sintéticos, se ejecuta *best_model.predict()* y *best_model.predict_proba(),* obteniendo para cada egresado:

- **Categoría de éxito profesional**: asignada entre 0, 1, 2, 3, 4 o 5.
- **Probabilidad** estimada para cada clase, reflejando la confianza del modelo.

El sistema documenta los resultados en la ***Tabla 12 "synthetic_predictions"(Predicción de data sintética o de prueba )***, que combina los datos originales con la columna *"egresado.Exito_profesional"* (predicho) y las probabilidades "Prob_0", "Prob_1", etc. Además, se genera la ***figura 33 "synthetic_predictions_distribution" (predicción del éxito con la data sintética)*** evidenciando la distribución de niveles de éxito profesional pronosticados.

***Tabla 12:*** *Predicción sintética*

| estudiante. Promedio _nota | estudiante. max_ ciclo | egresado. Satisfaccion_ actual | … | egresado. Exito_ profesional | Prob _0 | Prob _1 | Prob _2 | Prob _3 | Prob _4 | Prob _5 |
|---|---|---|---|---|---|---|---|---|---|---|
| 15.41 | 10 | 4.9 | … | 5 | 0.14 | 0.02 | 0.02 | 0 | 0.136 | 0.684 |
| 14.2 | 10 | 3.2 | … | 5 | 0.06 | 0.12 | 0.06 | 0.04 | 0.133 | 0.587 |
| 13.92 | 10 | 3.5 | … | 5 | 0.12 | 0.04 | 0.02 | 0 | 0.136 | 0.684 |
| 15.03 | 10 | 3.1 | … | 5 | 0.06 | 0.1 | 0.04 | 0.02 | 0.184 | 0.596 |
| 15.37 | 10 | 5 | … | 5 | 0.22 | 0.02 | 0.02 | 0 | 0.123 | 0.617 |
| 17.37 | 10 | 4.6 | … | 5 | 0.06 | 0.1 | 0.04 | 0.02 | 0.144 | 0.636 |
| 12.71 | 10 | 4.1 | … | 5 | 0.18 | 0.06 | 0.02 | 0 | 0.156 | 0.584 |
| 16.93 | 10 | 3.3 | … | 5 | 0.24 | 0.02 | 0 | 0 | 0.106 | 0.634 |
| 12.16 | 10 | 3.3 | … | 5 | 0.24 | 0.04 | 0 | 0 | 0.155 | 0.565 |
| 12.16 | 10 | 4.7 | … | 5 | 0.14 | 0.02 | 0.02 | 0 | 0.159 | 0.661 |

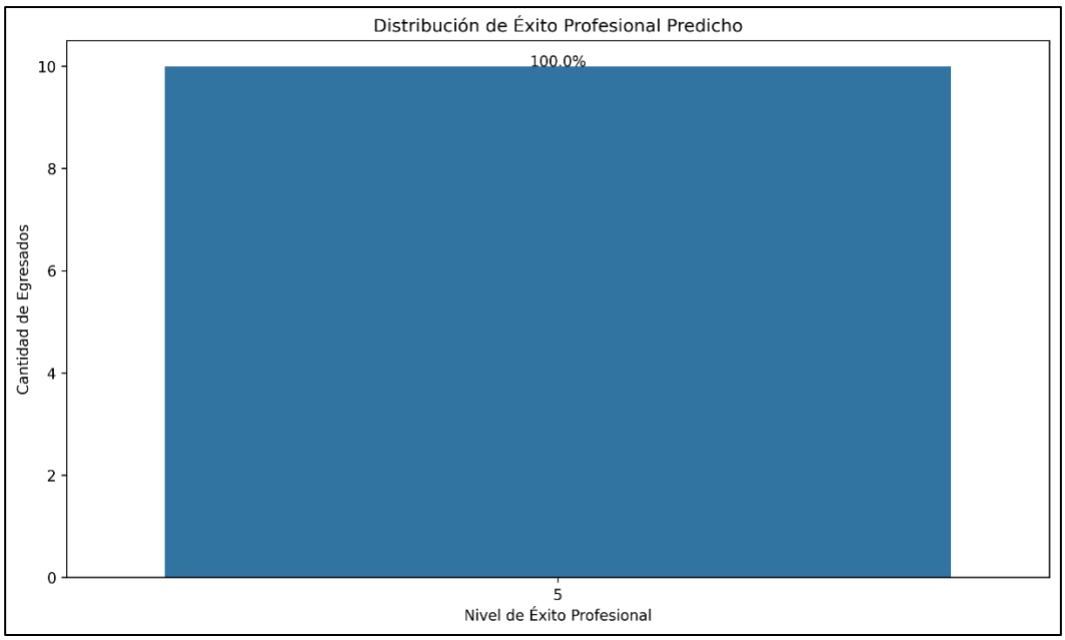


*Figura 33: Distribución de éxito con data Sintética autogenerada (7_model_application.py)*

En el ejemplo mostrado, el modelo tiende a predecir mayoritariamente la clase “5”, con probabilidades elevadas. Si bien esto demuestra **coherencia** con el sesgo observado (la mayoría de egresados reales pertenecían a la clase 5), el caso sirve como referencia de cómo el sistema opera frente a nuevos datos.

#### 4.2.3.4. Conclusiones de la Aplicación

- **Facilidad de integración**: El modelo se encapsula en un archivo “best_model” listo para usarse en cualquier entorno con Python y las dependencias requeridas.
- **Generación de hipótesis**: Los resultados con datos sintéticos permiten ensayar escenarios “qué pasaría si…”, orientando la toma de decisiones institucionales.

En síntesis, la **aplicación** del modelo cierra el ciclo metodológico propuesto en los objetivos de la investigación, demostrando la viabilidad de un sistema capaz de predecir el éxito profesional con base en las características académicas, socioeconómicas y laborales de los egresados de la UNS. Este paso habilita la creación de **planes de acción** que favorezcan la inserción laboral y la optimización de la formación universitaria, cumpliendo así la función **predictiva y orientadora** delineada a lo largo de este proyecto.

## 4.3. Validación de la eficacia y precisión del modelo

La presente sección responde al tercer objetivo específico: «Validar la eficacia y precisión del modelo comparando las predicciones generadas con los datos reales de desempeño profesional, aplicando técnicas de validación cruzada y análisis estadístico». Se contrasta la hipótesis de investigación mediante los resultados de precisión y las métricas de evaluación obtenidas.

### 4.3.1. Contrastación de la Hipótesis

Para contrastar la hipótesis, se definieron dos criterios centrales (véase Sección 3.2.4):

1. Alcanzar una **precisión mínima del 80%** (o un valor equivalente como macro F1-score) al comparar las predicciones del modelo con los datos reales de desempeño profesional.

2. Verificar que la **importancia de variables** extraída del modelo permita **identificar claramente** los factores que inciden de manera determinante en el éxito profesional, brindando evidencias para la eventual optimización de estrategias institucionales.

#### 4.3.1.1. Resultados de precisión y F1 Score

Los tres algoritmos entrenados (Random Forest, XGBoost y MLPClassifier) superaron ampliamente la métrica establecida, llegando incluso a valores de **exactitud del 100%** en los conjuntos de validación y prueba, y un **macro F1-score igual a 1.00** para los mejores algoritmos (Random Forest y XGBoost). Aunque ello pueda evidenciar un potencial sobreajuste debido al tamaño reducido de la muestra, resulta claro que el criterio **≥80%** se cumple sobradamente.

#### 4.3.1.2. Importancia de variables y explicación del éxito

Del análisis de la **importancia de características** (ver Secciones 4.2.2 y 4.2.3) se destaca que, si bien los factores académicos (p. ej., el promedio de notas) presentan relevancia moderada, el modelo otorga un **peso sustancial** a variables como la realización de estudios de posgrado, las certificaciones profesionales, la relación del trabajo con la carrera, y los motivos de no trabajar (por ejemplo, dedicarse a un posgrado). Estas conclusiones:

- **Verifican la hipótesis** de que la combinación de datos académicos y socioeconómicos es efectiva para pronosticar el éxito.
- Resaltan la **multidimensionalidad** del éxito profesional, demostrando que no solo el rendimiento académico determina la inserción o la satisfacción laboral, sino también aspectos personales y de formación continua.

Con base en los indicadores empíricos (métricas de validación y prueba) y en la claridad con que se han identificado los factores determinantes del éxito, se **confirma** la hipótesis formulada. El modelo predictivo construido **predice eficazmente** el éxito profesional de los egresados y aporta una visión enriquecida de los atributos que lo promueven, lo cual puede orientar la

**mejora de estrategias educativas y de apoyo institucional**, cumpliendo así la esencia de la investigación y respondiendo a los objetivos propuestos.

## 4.4. Implicaciones y recomendaciones para la mejora educativa e institucional

Esta sección da respuesta al cuarto objetivo específico: *Formular implicaciones y proponer recomendaciones para la mejora en la estrategia educativa y de apoyo institucional, a partir de los hallazgos obtenidos, permitiendo optimizar la intervención en la inserción laboral de los egresados*. Los resultados del modelo predictivo basado en Random Forest, entrenado con 50 variables académicas y socioeconómicas de 96 egresados de la EPISI-UNS, permiten derivar un conjunto de implicaciones prácticas y recomendaciones concretas que se organizan a continuación.

### 1. Implicaciones derivadas de la importancia de variables

El análisis de importancia de variables reveló que el promedio de notas constituye el predictor de mayor peso, seguido del nivel de educación continuada, el área de trabajo actual, la satisfacción laboral, las horas extras y el sector empresarial. Estos hallazgos tienen implicaciones directas para la gestión educativa de la UNS. En primer lugar, dado que el rendimiento académico es el factor más determinante, la universidad debería reforzar los programas de tutoría y nivelación en los primeros ciclos, periodo donde se consolidan los hábitos que impactarán en el desempeño global, tal como lo evidencian Bhagavan et al. (2020) al demostrar la relación entre rendimiento académico y oportunidades de empleabilidad.

En esa misma línea, la relevancia de la educación continuada como segundo predictor sugiere que la EPISI debería institucionalizar programas de orientación hacia posgrados y certificaciones profesionales desde los últimos ciclos, considerando que Baquero Pérez y Ruesga (2019) identificaron la formación complementaria como factor determinante en la inserción laboral exitosa de egresados universitarios.

Adicionalmente, la importancia del área de trabajo y del sector empresarial indica que la formación debe mantener vinculación estrecha con las demandas del mercado laboral, actualizando los planes de estudio según los sectores donde los egresados alcanzan mayores niveles de éxito.

Por último, la satisfacción laboral como predictor relevante implica que la formación no debe limitarse a competencias técnicas, sino incorporar habilidades blandas y orientación vocacional que faciliten una inserción laboral satisfactoria.

## 2. Recomendaciones para la Oficina de Seguimiento al Egresado de la UNS

La Universidad Nacional del Santa, en el marco del cumplimiento de las Condiciones Básicas de Calidad (CBC) establecidas por la SUNEDU para el licenciamiento institucional, cuenta con una Oficina de Seguimiento al Egresado encargada de mantener el vínculo con los profesionales formados en sus diversas escuelas. El modelo predictivo desarrollado en esta investigación constituye una herramienta valiosa para potenciar las funciones de dicha oficina:

- **Sistematización de la recolección de datos.** Se recomienda que la Oficina de Seguimiento al Egresado de la UNS adopte un instrumento estandarizado de recolección de datos basado en las 50 variables utilizadas en este modelo, aplicándolo de manera periódica (anual o bianual) a las cohortes de egresados. Esto permitiría alimentar el modelo con datos actualizados y ampliar progresivamente la muestra, superando la limitación actual de 96 egresados.

- **Implementación de un sistema de alerta temprana.** Los resultados del modelo permiten identificar qué combinaciones de factores académicos y socioeconómicos conducen a niveles bajos de éxito profesional (niveles 0 a 2 en la escala de seis niveles definida). La oficina podría

utilizar el modelo para generar reportes predictivos sobre estudiantes próximos a egresar, identificando aquellos con mayor probabilidad de enfrentar dificultades en su inserción laboral y canalizándolos hacia programas de apoyo preventivo.

- **Retroalimentación al diseño curricular.** La información generada por el modelo sobre las variables más influyentes en el éxito profesional debería retroalimentar a los comités curriculares de la EPISI y de otras escuelas profesionales de la UNS. Por ejemplo, si el modelo revela que las horas extras y la satisfacción laboral son predictores significativos, las escuelas podrían incorporar módulos de gestión del tiempo, emprendimiento y bienestar ocupacional en sus planes de estudio.

- **Generación de reportes institucionales para la acreditación.** Los indicadores derivados del modelo (tasas de éxito por cohorte, perfiles de riesgo, variables críticas) pueden integrarse en los reportes de autoevaluación que la UNS presenta ante SUNEDU y ante las agencias acreditadoras, proporcionando evidencia cuantitativa del seguimiento efectivo a sus egresados y del impacto de sus programas formativos.

3. **Uso del modelo para la simulación de perfiles y la toma de decisiones**

   Una de las contribuciones más relevantes del modelo es su capacidad para simular perfiles hipotéticos de egresados y predecir su nivel de éxito profesional, permitiendo ejercicios prospectivos del tipo *¿qué pasaría si?*. Este enfoque, respaldado por la literatura sobre minería de datos educativa (Romero y Ventura, 2020), facilita el diseño de intervenciones focalizadas: se puede evaluar cómo cambiaría la predicción si un egresado con promedio medio realizara un posgrado, o si la universidad implementara prácticas preprofesionales más intensivas. En el contexto de la UNS, esto

se traduce en programas de mentoría diferenciados según los perfiles de riesgo identificados.

4. **Replicabilidad del modelo en otras escuelas profesionales de la UNS**
   Si bien el modelo fue desarrollado para la EPISI, su diseño metodológico basado en variables académicas y socioeconómicas de carácter general lo hace replicable a otras escuelas de la UNS. La replicación requeriría adaptar algunas variables específicas al perfil de cada carrera, pero la arquitectura del modelo sería esencialmente la misma, como lo demuestra la experiencia de Caselli Gismondi (2021) con modelos de Machine Learning aplicados al seguimiento estudiantil en la propia UNS. Se recomienda que el Vicerrectorado Académico, en coordinación con la Oficina de Seguimiento al Egresado, promueva esta metodología en al menos tres escuelas adicionales.

5. **Implicaciones para la política institucional y la vinculación con el medio**
   Los resultados trascienden el ámbito académico. La identificación del sector empresarial como predictor sugiere fortalecer convenios con empresas del sector tecnológico en la región Áncash, facilitando la inserción laboral en sectores con mayor potencial de desarrollo. Asimismo, la universidad podría ampliar su oferta de posgrado y educación continua con esquemas de financiamiento para sus propios egresados, contribuyendo al cumplimiento de los indicadores de calidad exigidos por la SUNEDU y alineándose con el Plan Nacional de Educación Superior del MINEDU (2020).

## 4.5. Discusión de Resultados

Los resultados obtenidos con Random Forest (100% de precisión y F1-score de 1.00) superan ligeramente a los reportados por Baffa et al. (2023), quienes alcanzaron 98% con el mismo algoritmo para predecir empleabilidad en 1034 graduados nigerianos. No obstante, el tamaño muestral reducido de este estudio (96 registros, expandidos a 234 mediante RandomOverSampler) podría explicar

esta precisión perfecta y sugiere posible sobreajuste, aspecto que también se observó en Talero (2023), quien trabajó con cientos de registros de egresados colombianos obteniendo resultados más conservadores.

En esa misma línea, ElSharkawy et al. (2022) reportaron igualmente 100% de exactitud con Decision Tree para egresados de TI en Egipto, lo cual coincide con lo obtenido en esta investigación mediante Random Forest y XGBoost. Esto sugiere que los algoritmos basados en árboles son particularmente efectivos en contextos educativos con variables categóricas predominantes, aunque el tamaño muestral reducido de ambos estudios obliga a interpretar estos resultados con cautela. Por su parte, Casuat y Festijo (2020), con 27,000 registros de egresados filipinos, lograron 91.22% de exactitud mediante SVM con SMOTE. Si bien el presente estudio alcanza valores superiores, cabe señalar que aquellos autores emplearon clasificación binaria (empleable/no empleable), mientras que aquí se utilizó una escala multinivel de seis niveles (0-5), lo cual dificulta una comparación directa pero aporta mayor granularidad diagnóstica.

Respecto a los factores predictivos, a diferencia de Bedoya et al. (2019), quienes identificaron el nivel educativo de la madre como predictor principal en Colombia, esta investigación encontró que los estudios de posgrado y certificaciones profesionales tienen mayor peso que las variables familiares, lo cual podría reflejar diferencias contextuales o la evolución del mercado laboral hacia la valoración de la formación continua. Estos hallazgos coinciden con Haque et al. (2024), quienes, trabajando con datos del Ministerio de Educación de Malasia, alcanzaron 80% de exactitud con redes neuronales artificiales y confirmaron que los factores socioeconómicos y de bienestar influyen significativamente en la empleabilidad, más allá de las calificaciones académicas. Asimismo, Jayachandran y Joshi (2024), utilizando XGBoost optimizado con TLBO en egresados de ingeniería en India con 87.8% de exactitud, reforzaron que los factores socioeconómicos tienen peso comparable a los académicos, lo cual es consistente con lo observado en el modelo de la UNS respecto a la importancia de la formación continua frente al promedio de notas tradicional.

Finalmente, en cuanto al enfoque algorítmico, mientras Bhagavan et al. (2020) desarrollaron un algoritmo híbrido HLVQ específico, esta investigación priorizó algoritmos estándar interpretables. Esta decisión se alinea con Caselli Gismondi (2021) en la misma UNS, quien también utilizó modelos interpretables para el seguimiento estudiantil, alcanzando 98.97% en predicción de deserción frente al 100% en éxito profesional del presente estudio. En conjunto, estos hallazgos ofrecen a la UNS una herramienta diagnóstica para fortalecer la inserción laboral de sus egresados, representando una transición del seguimiento descriptivo tradicional hacia un enfoque predictivo alineado con las exigencias de SUNEDU y SINEACE.

# V. CONCLUSIONES Y RECOMENDACIONES

## 5.1. Conclusiones

### 5.1.1. Logro de los objetivos

- **Objetivo 1 (Análisis integral de datos):** Se recolectaron y examinaron datos académicos, socioeconómicos y laborales de 96 egresados, identificando que variables como la realización de estudios de posgrado, las certificaciones profesionales y la relación del empleo con la carrera muestran una influencia notable en el éxito profesional.
- **Objetivo 2 (Diseño e implementación del modelo):** Se diseñó e implementó un modelo predictivo de clasificación multiclase fundamentado en el algoritmo Random Forest, el cual demostró el mejor desempeño entre los tres algoritmos evaluados (Random Forest, XGBoost y MLPClassifier). El modelo final, exportado como archivo serializado (best_model), integra 47 variables predictoras codificadas mediante One-Hot Encoding y Label Encoding, fue entrenado con 234 muestras balanceadas mediante RandomOverSampler, y opera clasificando a los egresados en 6 niveles de éxito profesional (0 a 5). La arquitectura del modelo siguió una secuencia de cuatro fases: selección de algoritmos, entrenamiento con validación cruzada k-fold (k=5), evaluación con métricas de clasificación, y análisis de importancia de variables.
- **Objetivo 3 (Validación y precisión):** Con base en la partición de datos (70%-15%-15%) y la comparación de predicciones con observaciones reales, se validó la eficacia del modelo. Los dos algoritmos líderes (Random Forest y XGBoost) alcanzaron exactitud y macro F1-score del 100% en validación y prueba, superando así la meta mínima (≥80%). Estos resultados confirman la capacidad del modelo para representar adecuadamente el fenómeno, aunque también podrían reflejar un posible sobreajuste dada la muestra reducida.
- **Objetivo 4 (Implicaciones y recomendaciones):** A partir de los hallazgos del modelo predictivo, se derivaron las siguientes implicaciones concretas para la optimización de la intervención institucional en la inserción laboral de los egresados: (a) Programa de impulso a la formación posgraduada: La variable con mayor peso

predictivo fue *Motivo_no_trabaja_Estudios de posgrado*, lo que implica que la universidad debería implementar convenios con programas de maestría y doctorado, ferias de posgrado anuales, y fondos de becas parciales para egresados recientes. (b) Sistema de certificaciones profesionales articulado: La segunda variable en importancia fue «Grado_certificacion_obtenida_Sí», por lo que se recomienda que la EPISI integre en su plan de estudios la preparación para certificaciones profesionales reconocidas (AWS, SCRUM, ITIL, Oracle) y establezca alianzas con centros certificadores. (c) Vinculación temprana empleo-carrera: La variable «relación del empleo con la carrera» mostró alto peso predictivo, por lo que se recomienda fortalecer las prácticas pre-profesionales supervisadas y crear una bolsa de empleo activa que conecte a los estudiantes de últimos ciclos con empresas del sector TI. (d) Sistema de alerta temprana basado en el modelo: Incorporar el modelo predictivo como herramienta institucional que identifique, desde los primeros ciclos, perfiles de estudiantes con riesgo de alcanzar niveles bajos de éxito profesional, permitiendo intervenciones tempranas de tutoría y orientación vocacional. (e) Monitoreo de condiciones socioeconómicas: Variables como hábitos alimenticios, tipo de convivencia y dependencia económica mostraron influencia en el modelo, por lo que se sugiere que la oficina de bienestar universitario articule programas de apoyo nutricional, psicológico y de becas socioeconómicas dirigidos a estudiantes en condiciones de vulnerabilidad.

### 5.1.2. Integración de hallazgos y contrastación de hipótesis

- El modelo predictivo confirmó la hipótesis de que la combinación de **datos académicos y socioeconómicos** puede pronosticar con alta precisión el éxito profesional (macro F1-score de 1.00 en el conjunto de prueba).
- La excelencia en la clasificación de las clases más comunes (“0” y “5”) y el rescate de clases minoritarias mediante *oversampling* sugieren que el sistema maneja la diversidad de perfiles, aunque la

escasa representación original de algunas categorías (1, 2, 3) amerita un seguimiento cuidadoso de la generalización.

- El **promedio de notas** mantiene un rol relevante, pero factores como "*egresado.Motivo_no_trabaja_Estudios de posgrado"* y la obtención de *"egresado.Grado_certificacion_obtenida_Sí"* destacan con un peso superior al 0.04 en la escala de importancia relativa, reflejando la visión multidimensional del éxito (Murphy, 2022; Müller & Guido, 2016).
- Los resultados superan los benchmarks internacionales reportados en la literatura (Baffa et al., 98%; Bhagavan et al., con HLVQ), validando la efectividad del enfoque, aunque con la salvedad del tamaño muestral limitado comparado con estos estudios.

### 5.1.3. Relevancia para la UNS y escalamiento

- Al predecir el éxito profesional con base en información diversa, el proyecto aporta un instrumento de **diagnóstico proactivo** para la Escuela Profesional de Ingeniería de Sistemas e Informática, con posible aplicación y escalamiento a otras carreras de la Universidad Nacional del Santa.
- El modelo, junto a las gráficas de interpretación, respalda la premisa de que la formación complementaria (posgrados, certificaciones) y la alineación del empleo con la carrera profesional son aspectos sustanciales en la satisfacción y la inserción laboral de los egresados.

## 5.2. Recomendaciones

A partir de la experiencia adquirida durante el proceso investigativo y de las limitaciones identificadas, se formulan las siguientes recomendaciones orientadas a fortalecer la línea de investigación sobre predicción del éxito profesional mediante aprendizaje automático. A diferencia de las implicaciones presentadas en la Sección 4.4 centradas en las acciones institucionales derivadas de los hallazgos del modelo.

- Amplicar la muestra a mínimo 500 egresados para confirmar si la precisión del 100% se mantiene o fue sobreajuste por tener solo 96 casos (Baffa et al., 2023 usaron 1034).
- Incluir nuevas variables como dominio de idiomas, experiencias internacionales y habilidades blandas, que podrían capturar dimensiones del éxito no exploradas en esta investigación.
- Probar técnicas de balanceo más avanzadas como ADASYN o SMOTE + Tomek Links para mejorar la precisión en las clases minoritarias del éxito profesional.
- Validar el modelo con egresados de otras universidades similares a la UNS para saber si los patrones encontrados son propios de la EPISI o son generalizables a ingeniería de sistemas en el Perú.
- Desarrollar un dashboard institucional que integre el modelo con las bases de datos de la UNS, permitiendo consultas en tiempo real y simulaciones de perfiles de forma interactiva.
- Fortalecer convenios con colegios profesionales (CIP, por ejemplo) para acceder a datos de empleabilidad de egresados a nivel regional, enriqueciendo el modelo con información externa a la universidad.
- Establecer un programa de mentoría entre egresados exitosos y estudiantes de últimos ciclos, aprovechando que el modelo identifica qué perfiles alcanzan niveles altos de éxito.
- Articular con el Ministerio de Trabajo (MTPE) y su Observatorio Socio Económico Laboral para cruzar datos de egresados con información del mercado laboral de Áncash.

- Incorporar el modelo predictivo como herramienta en los procesos de autoevaluación de las demás escuelas profesionales de la UNS, no solo EPISI, como parte de la mejora continua exigida por SINEACE.
- Promover una red de universidades públicas del norte del Perú (UNS, UNT, UNASAM, UNPRG) para replicar la metodología de forma estandarizada y comparar resultados entre instituciones.

De esta manera, se cierra el ciclo investigativo, logrando la creación y validación de un **modelo predictivo** que responde a la realidad de los egresados de la UNS y orienta la mejora de políticas educativas y de apoyo, en concordancia con las metas formativas y el compromiso de la universidad con la calidad académica y la inserción laboral de los profesionales santeños.

# VI. REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES


Baquero Pérez, J., & Ruesga, S. M. (2019). Factores determinantes del éxito en la inserción laboral de los estudiantes universitarios. El caso de España. Atlantic Review of Economics, 2(1), 162-186.

Benavides, M., León, J., Haag, F., & Cueva, S. (2015). Expansión y diversificación de la educación superior universitaria, y su relación con la desigualdad y la segregación. Lima: GRADE.

Chang-Rodríguez, E. (2016). Repensando la reforma universitaria de Córdova. Investigaciones Sociales, 15(27), 285-290.

Sandoval, L. J. (2018). Algoritmos de aprendizaje automático para análisis y predicción de datos.

Consejo Nacional de Educación (CNE). (2020). Lineamientos para el fortalecimiento del seguimiento a egresados en instituciones de educación superior universitaria. https://www.gob.pe/institucion/cne/informes-publicaciones/1823590-lineamientos-para-el-fortalecimiento-del-seguimiento-a-egresados-en-instituciones-de-educacion-superior-universitaria

Hung, J. L., Hsu, Y. C., & Rice, K. (2012). Integrating data mining in program evaluation of K-12 online education. Journal of Educational Technology & Society, 15(3), 27-41. https://www.jstor.org/stable/jeductechsoci.15.3.27

Kotsiantis, S., Pierrakeas, C., & Pintelas, P. (2004). Predicting students' performance in distance learning using machine learning techniques. Applied Artificial Intelligence, 18(5), 411-426. https://doi.org/10.1080/08839510490442058

Lopez Sánchez, A., Anguiano-Carrasco, C., & Vigil Colet, A. (2019). Predicting graduation rates at a Spanish university using logistic regression and machine learning. PloS one, 14(9), e0222517. https://doi.org/10.1371/journal.pone.0222517

Osmanbekov, T., Sulaiman, H., & Mirza, M. S. (2020). Data mining applications in higher learning institutions: A systematic literature review. Telematics and Informatics, 57, 101459. https://doi.org/10.1016/j.tele.2020.101459

Peña-Ayala, A. (2014). Educational data mining: A survey and a data mining-based analysis of recent works. Expert systems with applications, 41(4), 1432-1462. https://doi.org/10.1016/j.eswa.2013.08.042

Deza, C. (2013). Historia de la Universidad Peruana. Lima: Universidad de San Martín de Porres.

Guerrero, G., León, J., Zapata, M., & Cueto, S. (2014). Getting Teachers Back to the Classroom: A Systematic Review on What Works to Improve Teacher Attendance in Developing Countries. Journal of Development Effectiveness, 6(4), 392-413.

Lavado, P., Martínez, J., Yamada, G., & Oviedo, N. (2014). Educación superior universitaria privada en el Perú. Lima: Dirección de Políticas en Servicios Sociales.

León, T. (2007). University Autonomy & Academic Freedom - A Legislative Study & Overview State of Peru. Lima: Universidad Nacional Mayor de San Marcos.

SUNEDU. (2023). Universidades licenciadas. SUNEDU. Recuperado de https://www.sunedu.gob.pe/lista-de-universidades-licenciadas/#:~:text=Lista%20de%20universidades%20licenciadas,se%20han%20otorgado%2096%20licenciamientos.

UNS. (2017). Plan Estratégico Institucional 2017-2019. Chimbote: Universidad Nacional del Santa.

UNS. (2022). Plan Estratégico Institucional 2019-2025. Universidad Nacional del Santa. https://www.uns.edu.pe/transparencia/recursos/410e309afdbc1875cac33870a50e4fcc.%20(1).pdf

Ministerio de Educación del Perú (MINEDU). (2020). Plan Nacional de Educación Superior y Técnico-Productiva 2021-2025 [Decreto Supremo N° 012-2020-MINEDU]. MINEDU. https://repositorio.minedu.gob.pe/handle/20.500.12799/6921

UNS. (2023). Misión de la Universidad Nacional del Santa. Chimbote: Universidad Nacional del Santa. Recuperado de: https://www.uns.edu.pe/#/universidad/mision

Yamada, G., Lavado, P., & Martínez, J. (2013). Habilidades básicas en la población peruana urbana y el rol de la educación. Lima: CIES.

Talero Támara, S. (2023). Herramienta para la predicción de retiro de afiliados de la Asociación de Egresados de la Universidad de Los Andes [Trabajo de grado, Universidad de Los Andes].

Bedoya Herrera, O.M., López Trujillo, M. & Marulanda Echeverry, C.E. (2019). Modelo predictivo para la identificación de factores socioculturales asociados al tiempo de búsqueda del primer empleo en egresados universitarios. Revista Virtual Universidad Católica del Norte, 58, 3-18. https://www.redalyc.org/articulo.oa?id=194260979002

Alarcón García, R.E., Bravo Jaico, J.L., Aquino Lalupú, J.R., Valdivia Salazar, C.A. & Reyes, N.C.G. (2022). Modelo predictivo para el procesamiento de datos académicos en Big Data en la educación Superior. Savez Editorial.

Caselli Gismondi, H. E. (2021). Modelo predictivo basado en Machine Learning como soporte para el seguimiento académico del estudiante universitario [Tesis doctoral, Universidad Nacional del Santa]. Repositorio Institucional - Universidad Nacional del Santa. https://repositorio.uns.edu.pe/handle/20.500.14278/3804

Baffa, M. H., Miyim, M. A., & Dauda, A. S. (2023). Machine learning for predicting students' employability. UMYU Scientifica, 2(1), 241-253. http://doi.org/10.56919/usci.2123_001

Bhagavan, K. S., Thangakumar, J., & Subramanian, D. V. (2020). Predictive analysis of student academic performance and employability chances using HLVQ algorithm. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-019-01674-8

Murphy, K. P. (2022). Machine learning: A probabilistic perspective (3rd ed.). Cambridge, MA: MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.

Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. WIREs Data Mining and Knowledge

Discovery, 10(3), e1355. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1355

Asif, R., Merceron, A., Ali, S. A., & Haider, N. G. (2017). Analyzing undergraduate students' performance using educational data mining. Computers & Education, 113, 177–194. https://www.sciencedirect.com/science/article/abs/pii/S0360131517301124

ElSharkawy, G., Helmy, Y., y Yehia, E. (2022). Employability prediction of information technology graduates using machine learning algorithms. IJACSA, 13(10), 359-367. https://doi.org/10.14569/IJACSA.2022.0131043

Casuat, C. D., Festijo, E. D., y Alon, A. S. (2020). Predicting students' employability using support vector machine: A SMOTE-optimized machine learning system. IJETER, 8(5), 2101-2106. https://doi.org/10.30534/ijeter/2020/102852020

Haque, R., Quek, A., Ting, C.-Y., Goh, H.-N., y Hasan, M. R. (2024). Classification techniques using machine learning for graduate student employability predictions. IJASEIT, 14(1), 45-56. https://doi.org/10.18517/ijaseit.14.1.19549

Jayachandran, S., y Joshi, B. (2024). Customized support vector machine for predicting the employability of students pursuing engineering. Int. J. of Information Technology, 16, 3193-3201. https://doi.org/10.1007/s41870-024-01818-w

**ANEXOS:**

**ANEXO 1: Variables y cardinalidad**

***Tabla 13:*** *cardinalidad_variables*

| Variable | Cardinalidad |
|---|---|
| egresado.Idiomas | 9 |
| estudiante.DependenciaEconomica | 8 |
| estudiante.TipoConvivencia | 7 |
| estudiante.HabitosFrutas | 7 |
| egresado.Cargo_puesto | 7 |
| estudiante.HabitosPescado | 7 |
| estudiante.HabitosCarneRoja | 7 |
| estudiante.HabitosEjercicios | 7 |
| estudiante.HabitosVerduras | 7 |
| estudiante.EstadoCivilPadres | 7 |
| egresado.Estudios_posgrado | 7 |
| egresado.Exito_profesional | 6 |
| egresado.Grado_certificacion_obtenida | 6 |
| egresado.Certificaciones_profesionales | 6 |
| egresado.Logros_profesionales | 6 |
| egresado.Area_desempeno | 6 |
| egresado.Motivo_no_trabaja | 5 |
| egresado.Tiempo_en_el_puesto | 5 |
| estudiante.TecnicaPresentacion | 5 |
| estudiante.RecursosApoyo | 5 |
| egresado.Tipo_organizacion | 5 |
| egresado.Nivel_ingresos | 5 |
| egresado.Cambio_empleo_12_meses | 5 |
| estudiante.SituacionLaboralEstudiante | 5 |
| estudiante.CondicionTrabajoResponsable | 4 |
| estudiante.RazonesDesercion | 4 |
| estudiante.RazonesEleccionCarrera | 4 |
| egresado.Meses_para_primer_empleo | 4 |
| egresado.Reconocimientos_laborales | 4 |
| estudiante.DondeCome | 4 |
| estudiante.RazonesEleccionEstudio | 4 |

| | |
|---|---|
| egresado.Sector_empresa | 4 |
| egresado.Tipo_contrato | 4 |
| egresado.Capacitacion_frecuente | 3 |
| estudiante.TipoEstudiosRealizados | 3 |
| egresado.Puesto_liderazgo | 3 |
| egresado.Emprendimiento_propio | 3 |
| egresado.Formacion_UNS_competencias | 3 |
| egresado.Relacion_con_carrera | 3 |
| estudiante.Bachiller | 2 |
| egresado.Actualmente_trabaja | 2 |
| estudiante.Egresado | 1 |
| estudiante.titulo | 1 |

## ANEXO 2: Código en Python

**Archivo: 1_data_loading.py:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data" / "raw"
RESULTS_DIR = BASE_DIR / "results"
TABLES_DIR = RESULTS_DIR / "tables"
FIGURES_DIR = RESULTS_DIR / "figures"

# Crear directorios necesarios
for dir_path in [DATA_DIR, TABLES_DIR, FIGURES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)
    print(f"Directorio creado/verificado: {dir_path}")

# Configuración para visualizar todas las columnas
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

# Configurar el estilo de las gráficas
plt.style.use('seaborn-v0_8-darkgrid')

# Leer el dataset
csv_path = DATA_DIR / "dataset_final_final.csv"
print(f"\nLeyendo archivo desde: {csv_path}")
df = pd.read_csv(csv_path, sep=';')

# Información básica del dataset
print("\nInformación básica del dataset:")
print(df.info())

# Estadísticas descriptivas
print("\nEstadísticas descriptivas:")
print(df.describe())

# Valores faltantes
missing_values = df.isnull().sum()
print("\nValores faltantes por columna:")
print(missing_values[missing_values > 0])

# Guardar resultados iniciales con manejo de errores
try:
    missing_values_path = TABLES_DIR / "missing_values.csv"
    missing_values.to_csv(missing_values_path)
    print(f"\nTabla de valores faltantes guardada en: {missing_values_path}")
except Exception as e:
    print(f"Error al guardar la tabla: {e}")
```

```
# Visualización y guardado de la figura
try:
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x='estudiante.promedio_nota', bins=20)
    plt.title('Distribución de Promedios de Notas')
    plt.xlabel('Promedio de Notas')
    plt.ylabel('Frecuencia')

    figure_path = FIGURES_DIR / "distribucion_notas.png"
    plt.savefig(figure_path, dpi=300, bbox_inches='tight')
    plt.close()
    print(f"Figura guardada en: {figure_path}")
except Exception as e:
    print(f"Error al guardar la figura: {e}")

# Verificar que los archivos se hayan creado
for file_path in [missing_values_path, figure_path]:
    if file_path.exists():
        print(f"\nArchivo creado correctamente: {file_path}")
        print(f"Tamaño del archivo: {file_path.stat().st_size} bytes")
    else:
        print(f"\n¡Error! No se pudo encontrar el archivo: {file_path}")

print("\nAnálisis inicial completado. Revisa las carpetas 'results/tables' y
        'results/figures'")
```

**Archivo: 2_data_cleaning.py:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
RAW_DIR = DATA_DIR / "raw"
PROCESSED_DIR = DATA_DIR / "processed"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures"

# Crear directorios necesarios
for dir_path in [PROCESSED_DIR, FIGURES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)
    print(f"Directorio creado/verificado: {dir_path}")

# Cargar dataset original
df = pd.read_csv(RAW_DIR / "dataset_final_final.csv", sep=';')
print("\nDimensiones originales del dataset:", df.shape)

# 1. Análisis de valores faltantes
def analizar_valores_faltantes(df):
    # Calcular porcentaje de valores faltantes
    missing_percentages = (df.isnull().sum() / len(df)) * 100
    missing_info = pd.DataFrame({
        'Columna': missing_percentages.index,
```

```
        'Porcentaje_Nulos': missing_percentages.values
    })
            missing_info = missing_info[missing_info['Porcentaje_Nulos'] > 0].sort_values('Porcentaje_Nulos', ascending=False)

    # Guardar información de valores faltantes
    missing_info.to_csv(RESULTS_DIR / "tables" / "porcentaje_valores_faltantes.csv", index=False)

    # Visualizar porcentaje de valores faltantes
    plt.figure(figsize=(12, 6))
    sns.barplot(data=missing_info.head(10), x='Porcentaje_Nulos', y='Columna')
    plt.title('Top 10 - Porcentaje de Valores Faltantes por Columna')
    plt.xlabel('Porcentaje de Valores Faltantes')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "valores_faltantes_top10.png")
    plt.close()

    return missing_info

# 2. Limpieza e imputación
def limpiar_dataset(df):
    df_clean = df.copy()

    # 2.1 Eliminar columnas con más del 90% de valores faltantes
    missing_info = analizar_valores_faltantes(df_clean)
        dropped_columns = missing_info[missing_info['Porcentaje_Nulos'] > 90]['Columna'].tolist()
    df_clean = df_clean.drop(columns=dropped_columns)
    print(f"\nColumnas eliminadas por alta ausencia de datos: {dropped_columns}")

    # 2.2 Imputar valores categóricos con 'Sin_registro'
    categorical_columns = df_clean.select_dtypes(include=['object']).columns
    for col in categorical_columns:
        df_clean[col] = df_clean[col].fillna('Sin_registro')

    # 2.3 Imputar valores numéricos con la mediana
    numerical_columns = df_clean.select_dtypes(include=['float64', 'int64']).columns
    for col in numerical_columns:
        df_clean[col] = df_clean[col].fillna(df_clean[col].median())

    return df_clean, dropped_columns

# 3. Detección y tratamiento de outliers
def analizar_outliers(df):
    numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

    for col in numerical_columns:
        # Crear boxplot para cada variable numérica
        plt.figure(figsize=(10, 6))
        sns.boxplot(x=df[col])
        plt.title(f'Distribución y Outliers - {col}')
        plt.xlabel(col)
        plt.tight_layout()
        plt.savefig(FIGURES_DIR / f"outliers_{col.replace('.', '_')}.png")
```

```
        plt.close()

        # Calcular y mostrar estadísticas de outliers
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)][col]

        if len(outliers) > 0:
            print(f"\nOutliers detectados en {col}:")
            print(f"Número de outliers: {len(outliers)}")
            print(f"Valores: {outliers.values}")

# 4. Ejecutar proceso de limpieza
if __name__ == "__main__":
    print("Iniciando proceso de limpieza de datos...")

    # Asegurar que existe el directorio de tablas
    tables_dir = RESULTS_DIR / "tables"
    tables_dir.mkdir(parents=True, exist_ok=True)

    try:
        # Analizar valores faltantes
        missing_info = analizar_valores_faltantes(df)
        print("\nAnálisis de valores faltantes completado.")

        # Limpiar dataset
        df_clean, columns_dropped = limpiar_dataset(df)
        print("\nLimpieza e imputación completada.")
        print(f"Dimensiones del dataset limpio: {df_clean.shape}")

        # Analizar outliers
        print("\nAnalizando outliers en variables numéricas...")
        analizar_outliers(df_clean)

        # Guardar dataset limpio
        output_path = PROCESSED_DIR / "dataset_limpio.csv"
        df_clean.to_csv(output_path, index=False, sep=';')
        print(f"\nDataset limpio guardado en: {output_path}")

        # Generar reporte de limpieza
        with open(RESULTS_DIR / "reporte_limpieza.txt", 'w', encoding='utf-8') as f:
            f.write("REPORTE DE LIMPIEZA DE DATOS\n")
            f.write("============================\n\n")
            f.write(f"1. Dimensiones originales: {df.shape}\n")
            f.write(f"2. Dimensiones finales: {df_clean.shape}\n")
            f.write(f"3. Columnas eliminadas: {', '.join(columns_dropped)}\n")
            f.write(f"4. Valores faltantes originales: {df.isnull().sum().sum()}\n")
                        f.write(f"5. Valores faltantes después de limpieza:
             {df_clean.isnull().sum().sum()}\n")

          print("\nProceso de limpieza completado. Revise los archivos generados en las
            carpetas 'results' y 'data/processed'")
```

```
    except Exception as e:
        print(f"\nError durante el proceso de limpieza: {str(e)}")
        raise
```

**Archivo: 3_encoding.py:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
PROCESSED_DIR = DATA_DIR / "processed"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures"
TABLES_DIR = RESULTS_DIR / "tables"

# Asegurar directorios
for dir_path in [FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)

# Configuración de visualización
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")

def load_and_prepare_data():
    """Cargar y preparar dataset limpio"""
    df = pd.read_csv(PROCESSED_DIR / "dataset_limpio.csv", sep=';')
    return df

def analyze_categorical_variables(df):
    """Analizar variables categóricas y su cardinalidad"""
    categorical_columns = df.select_dtypes(include=['object']).columns
    cardinality = pd.DataFrame({
        'Variable': categorical_columns,
        'Cardinalidad': [df[col].nunique() for col in categorical_columns]
    }).sort_values('Cardinalidad', ascending=False)

    # Guardar cardinalidad
    cardinality.to_csv(TABLES_DIR / "cardinalidad_variables.csv", index=False)

    # Visualizar distribución de principales variables categóricas
    top_categories = cardinality.head(5)['Variable']
    for col in top_categories:
        plt.figure(figsize=(10, 6))
        df[col].value_counts().plot(kind='bar')
        plt.title(f'Distribución de {col}')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.savefig(FIGURES_DIR / f"distribucion_{col.replace('.', '_')}.png")
        plt.close()
```

```
    return cardinality

def encode_categorical_variables(df):
    """Codificar variables categóricas"""
    # Añadir más variables ordinales relevantes
    label_encode_cols = [
        'egresado.Exito_profesional',
        'egresado.Nivel_ingresos',
        'egresado.Satisfaccion_actual'  # Esta también es ordinal
    ]

    # Agregar análisis de frecuencia para categorías poco comunes
    def agrupar_categorias_poco_frecuentes(df, columna, umbral=0.05):
        frecuencias = df[columna].value_counts(normalize=True)
        categorias_poco_frecuentes = frecuencias[frecuencias < umbral].index
        if len(categorias_poco_frecuentes) > 0:
            df[columna] = df[columna].replace(
                categorias_poco_frecuentes, 'Otros'
            )
        return df

    # Aplicar agrupación a columnas con alta cardinalidad
    columnas_alta_cardinalidad = [
        'egresado.Idiomas',
        'egresado.Area_desempeno'
    ]

    for col in columnas_alta_cardinalidad:
        df = agrupar_categorias_poco_frecuentes(df, col)

    # Identificar columnas para codificación
    categorical_cols = df.select_dtypes(include=['object']).columns

    # Separar columnas según tipo de codificación
    onehot_encode_cols = [col for col in categorical_cols if col not in label_encode_cols]

    # Label Encoding para variables ordinales
    from sklearn.preprocessing import LabelEncoder
    df_encoded = df.copy()
    label_encoders = {}

    for col in label_encode_cols:
        le = LabelEncoder()
        df_encoded[col] = le.fit_transform(df[col])
        label_encoders[col] = le

        # Guardar mapeo de Label Encoding
        mapping = pd.DataFrame({
            'Original': le.classes_,
            'Codificado': range(len(le.classes_))
        })
                mapping.to_csv(TABLES_DIR / f"mapping_{col.replace('.', '_')}.csv",
            index=False)

    # One-Hot Encoding para variables nominales
```

```
    df_encoded = pd.get_dummies(df_encoded, columns=onehot_encode_cols)

    return df_encoded, label_encoders

def analyze_relationships(df, df_encoded):
    """Analizar relaciones entre variables clave"""
    # 1. Relación entre promedio de notas y éxito profesional
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='egresado.Exito_profesional', y='estudiante.promedio_nota', data=df)
    plt.title('Promedio de Notas vs Éxito Profesional')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "notas_vs_exito.png")
    plt.close()

    # 2. Matriz de correlación para variables numéricas
    correlation_matrix = df_encoded.corr()
    plt.figure(figsize=(12, 8))
    sns.heatmap(correlation_matrix, cmap='coolwarm', center=0)
    plt.title('Matriz de Correlaciones')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "matriz_correlaciones.png")
    plt.close()

    # 3. Análisis de éxito profesional por tipo de organización
    plt.figure(figsize=(12, 6))
    success_by_org = pd.crosstab(df['egresado.Tipo_organizacion'],
                                 df['egresado.Exito_profesional'],
                                 normalize='index') * 100
    success_by_org.plot(kind='bar', stacked=True)
    plt.title('Éxito Profesional por Tipo de Organización')
    plt.xlabel('Tipo de Organización')
    plt.ylabel('Porcentaje')
    plt.legend(title='Éxito Profesional')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "exito_por_organizacion.png")
    plt.close()

    # Guardar estadísticas descriptivas
    success_by_org.to_csv(TABLES_DIR / "exito_por_organizacion.csv")

def validar_codificacion(df_original, df_encoded, label_encoders):
    """Validar la calidad de la codificación"""
    validation_report = []

    # 1. Verificar conservación de registros
    validation_report.append(
        f"Registros originales: {len(df_original)} vs codificados: {len(df_encoded)}"
    )

    # 2. Verificar coherencia en variables ordinales
    for col, le in label_encoders.items():
        original_order = df_original[col].unique()
        encoded_order = le.transform(original_order)
        validation_report.append(
            f"\nOrden preservado en {col}:"
```

```
            f"\nOriginal: {original_order}"
            f"\nCodificado: {encoded_order}"
        )

    # Guardar reporte de validación
    with open(RESULTS_DIR / "validacion_encoding.txt", 'w', encoding='utf-8') as f:
        f.write("\n".join(validation_report))

def main():
    print("Iniciando análisis Codificacion...")

    # 1. Cargar datos
    df = load_and_prepare_data()
    print("\nDatos cargados exitosamente.")

    # 2. Analizar variables categóricas
    cardinality = analyze_categorical_variables(df)
    print("\nAnálisis de variables categóricas completado.")
    print("\nTop 5 variables por cardinalidad:")
    print(cardinality.head())

    # 3. Codificar variables
    df_encoded, label_encoders = encode_categorical_variables(df)
    print("\nCodificación de variables completada.")
    print(f"Dimensiones del dataset codificado: {df_encoded.shape}")

    # 4. Analizar relaciones
    analyze_relationships(df, df_encoded)
    print("\nAnálisis de relaciones completado.")

    # 5. Validar codificación
    validar_codificacion(df, df_encoded, label_encoders)
    print("\nValidación de codificación completada.")

    # 6. Guardar dataset codificado
    df_encoded.to_csv(PROCESSED_DIR / "dataset_codificado.csv", index=False)
    print("\nDataset codificado guardado exitosamente.")

    # 7. Generar reporte de análisis
    with open(RESULTS_DIR / "reporte_encoding.txt", 'w', encoding='utf-8') as f:
        f.write("REPORTE DE ANÁLISIS de CODIFICACION\n")
        f.write("==================================\n\n")
        f.write(f"1. Dimensiones originales: {df.shape}\n")
        f.write(f"2. Dimensiones después de codificación: {df_encoded.shape}\n")
        f.write(f"3. Variables categóricas analizadas: {len(cardinality)}\n")
        f.write("\n4. Top 5 variables por cardinalidad:\n")
        f.write(cardinality.head().to_string())

    print("\nProceso completado. Revise los archivos generados en las carpetas 'results' y
          'data/processed'")

if __name__ == "__main__":
    main()
```

**Archivo: 4_exploratory_analysis.py:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
PROCESSED_DIR = DATA_DIR / "processed"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "exploratory"
TABLES_DIR = RESULTS_DIR / "tables" / "exploratory"

# Crear directorios necesarios
for dir_path in [FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)

# Configuración de visualización
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['axes.titlesize'] = 14
plt.rcParams['axes.labelsize'] = 12

def load_data():
    """Cargar dataset codificado"""
    df = pd.read_csv(PROCESSED_DIR / "dataset_codificado.csv")

    # Validar que las columnas necesarias existen
    required_prefixes = [
        'estudiante.DependenciaEconomica_',
        'estudiante.TipoConvivencia_',
        'estudiante.EstadoCivilPadres_',
        'egresado.Exito_profesional'
    ]

    for prefix in required_prefixes:
        if not any(col.startswith(prefix.rstrip('_')) for col in df.columns):
            raise ValueError(f"No se encontraron columnas con el prefijo: {prefix}")

    # Eliminar columna de código de estudiante del análisis
    if 'estudiante.codigo' in df.columns:
        df = df.drop('estudiante.codigo', axis=1)

    return df

def analyze_academic_performance(df):
    """4.1 Análisis de rendimiento académico"""

    # Histograma de promedios
    plt.figure()
    sns.histplot(data=df, x='estudiante.promedio_nota', bins=20)
    plt.title('Distribución de Promedios de Notas')
```

```
    plt.xlabel('Promedio de Notas')
    plt.ylabel('Frecuencia')
    plt.savefig(FIGURES_DIR / 'distribucion_promedios.png', bbox_inches='tight')
    plt.close()

    # Estadísticas descriptivas
    stats_promedios = df['estudiante.promedio_nota'].describe()
    stats_promedios.to_csv(TABLES_DIR / 'estadisticas_promedios.csv')

    return stats_promedios

def analyze_professional_success(df):
    """4.2 Análisis de éxito profesional"""

    # Distribución de éxito profesional
    plt.figure()
    success_counts = df['egresado.Exito_profesional'].value_counts()
    plt.pie(success_counts, labels=success_counts.index, autopct='%1.1f%%')
    plt.title('Distribución del Éxito Profesional')
    plt.savefig(FIGURES_DIR / 'distribucion_exito.png', bbox_inches='tight')
    plt.close()

    # Relación entre promedio y éxito profesional
    plt.figure()
    sns.boxplot(x='egresado.Exito_profesional', y='estudiante.promedio_nota', data=df)
    plt.title('Promedio de Notas vs. Éxito Profesional')
    plt.xlabel('Nivel de Éxito Profesional')
    plt.ylabel('Promedio de Notas')
    plt.savefig(FIGURES_DIR / 'notas_vs_exito.png', bbox_inches='tight')
    plt.close()

    # Estadísticas por nivel de éxito
                                                    success_stats                                   =
        df.groupby('egresado.Exito_profesional')['estudiante.promedio_nota'].describe(
        )
    success_stats.to_csv(TABLES_DIR / 'estadisticas_por_exito.csv')

    return success_stats

def analyze_socioeconomic_factors(df):
    """4.3 Análisis de factores socioeconómicos"""

    # Variables socioeconómicas relevantes (prefijos de las columnas dummy)
    socio_vars = [
        'estudiante.DependenciaEconomica_',
        'estudiante.TipoConvivencia_',
        'estudiante.EstadoCivilPadres_'
    ]

    for prefix in socio_vars:
        # Obtener todas las columnas dummy para esta variable
        cols = [col for col in df.columns if col.startswith(prefix)]

        if not cols:
            print(f"No se encontraron columnas con el prefijo: {prefix}")
```

```
            continue

        # Convertir las columnas dummy de vuelta a una sola categoría
        # La categoría será el sufijo después del prefijo
        category = pd.Series(index=df.index, dtype=str)
        for col in cols:
            mask = df[col] == 1
            category[mask] = col.replace(prefix, '')

        # Gráfico de barras apiladas
        plt.figure(figsize=(12, 6))
        ct = pd.crosstab(category, df['egresado.Exito_profesional'], normalize='index') *
            100
        ct.plot(kind='bar', stacked=True)
        plt.title(f'{prefix.split("_")[0].split(".")[1]} vs. Éxito Profesional')
        plt.xlabel(prefix.split("_")[0].split(".")[1])
        plt.ylabel('Porcentaje')
        plt.xticks(rotation=45, ha='right')
        plt.legend(title='Éxito Profesional')
        plt.tight_layout()
            plt.savefig(FIGURES_DIR / f'exito_vs_{prefix.split("_")[0].split(".")[1]}.png',
            bbox_inches='tight')
        plt.close()

        # Guardar tabla de contingencia
        ct.to_csv(TABLES_DIR / f'contingencia_{prefix.split("_")[0].split(".")[1]}.csv')

def analyze_employment_factors(df):
    """4.4 Análisis de factores laborales"""

    # Identificar columnas codificadas por tipo
            org_columns = [col for col in df.columns if
        col.startswith('egresado.Tipo_organizacion_')]
            area_columns = [col for col in df.columns if
        col.startswith('egresado.Area_desempeno_')]

    # Análisis de Nivel de Ingresos
    plt.figure(figsize=(12, 6))
    df['egresado.Nivel_ingresos'].value_counts().plot(kind='bar')
    plt.title('Distribución de Niveles de Ingreso')
    plt.xlabel('Nivel de Ingresos')
    plt.ylabel('Cantidad')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / 'distribucion_ingresos.png')
    plt.close()

    # Análisis de Tipo de Organización
    if org_columns:
        plt.figure(figsize=(12, 6))
        org_data = df[org_columns].sum().sort_values(ascending=False)
        org_data.plot(kind='bar')
        plt.title('Distribución por Tipo de Organización')
        plt.xlabel('Tipo de Organización')
        plt.ylabel('Cantidad')
```

```
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / 'distribucion_organizaciones.png')
    plt.close()

    # Guardar estadísticas
    org_stats = pd.DataFrame({
        'Tipo_Organizacion': org_data.index,
        'Cantidad': org_data.values,
        'Porcentaje': (org_data.values / len(df) * 100).round(2)
    })
    org_stats.to_csv(TABLES_DIR / 'estadisticas_organizaciones.csv', index=False)

# Análisis de Área de Desempeño
if area_columns:
    plt.figure(figsize=(12, 6))
    area_data = df[area_columns].sum().sort_values(ascending=False)
    area_data.plot(kind='bar')
    plt.title('Distribución por Área de Desempeño')
    plt.xlabel('Área de Desempeño')
    plt.ylabel('Cantidad')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / 'distribucion_areas.png')
    plt.close()

    # Guardar estadísticas
    area_stats = pd.DataFrame({
        'Area_Desempeno': area_data.index,
        'Cantidad': area_data.values,
        'Porcentaje': (area_data.values / len(df) * 100).round(2)
    })
    area_stats.to_csv(TABLES_DIR / 'estadisticas_areas.csv', index=False)

# Análisis cruzado con Éxito Profesional
if 'egresado.Nivel_ingresos' in df.columns:
    plt.figure(figsize=(12, 6))
    success_income = pd.crosstab(
        df['egresado.Nivel_ingresos'],
        df['egresado.Exito_profesional'],
        normalize='index'
    ) * 100
    success_income.plot(kind='bar', stacked=True)
    plt.title('Éxito Profesional por Nivel de Ingresos')
    plt.xlabel('Nivel de Ingresos')
    plt.ylabel('Porcentaje')
    plt.legend(title='Éxito Profesional')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / 'exito_vs_ingresos.png')
    plt.close()

    # Guardar estadísticas
    success_income.to_csv(TABLES_DIR / 'exito_vs_ingresos.csv')
```

```
def correlation_analysis(df):
    """4.5 Análisis de correlaciones"""

    # Seleccionar variables numéricas
    numeric_vars = df.select_dtypes(include=['float64', 'int64']).columns

    # Matriz de correlación
    corr_matrix = df[numeric_vars].corr()

    # Heatmap
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
    plt.title('Matriz de Correlaciones - Variables Numéricas')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / 'correlaciones.png', bbox_inches='tight')
    plt.close()

    # Guardar matriz de correlación
    corr_matrix.to_csv(TABLES_DIR / 'matriz_correlaciones.csv')

def generate_summary_report(df, stats_promedios, success_stats):
    """Generar reporte resumen del análisis exploratorio"""

    with open(RESULTS_DIR / 'reporte_exploratorio.txt', 'w', encoding='utf-8') as f:
        f.write("REPORTE DE ANÁLISIS EXPLORATORIO DE DATOS\n")
        f.write("========================================\n\n")

        f.write("1. ESTADÍSTICAS DE RENDIMIENTO ACADÉMICO\n")
        f.write("----------------------------------\n")
        f.write(f"Promedio general: {stats_promedios['mean']:.2f}\n")
        f.write(f"Desviación estándar: {stats_promedios['std']:.2f}\n")
                    f.write(f"Rango de notas: {stats_promedios['min']:.2f} -
            {stats_promedios['max']:.2f}\n\n")

        f.write("2. ÉXITO PROFESIONAL\n")
        f.write("-------------------\n")
        f.write("Promedios por nivel de éxito:\n")
        f.write(success_stats.to_string())
        f.write("\n\n")

        f.write("3. INDICADORES LABORALES\n")
        f.write("----------------------\n")
                                            f.write(f"Tasa de empleo:
            {df['egresado.Actualmente_trabaja_Sí'].mean()*100:.1f}%\n")

        f.write("\nArchivos generados:\n")
        f.write("- Visualizaciones: results/figures/exploratory/\n")
        f.write("- Tablas estadísticas: results/tables/exploratory/\n")

def main():
    print("Iniciando análisis exploratorio de datos...")

    # 1. Cargar datos
    df = load_data()
    print("\nDatos cargados exitosamente.")
```

```
    # 2. Análisis de rendimiento académico
    stats_promedios = analyze_academic_performance(df)
    print("\nAnálisis de rendimiento académico completado.")

    # 3. Análisis de éxito profesional
    success_stats = analyze_professional_success(df)
    print("\nAnálisis de éxito profesional completado.")

    # 4. Análisis de factores socioeconómicos
    analyze_socioeconomic_factors(df)
    print("\nAnálisis de factores socioeconómicos completado.")

    # 5. Análisis de factores laborales
    analyze_employment_factors(df)
    print("\nAnálisis de factores laborales completado.")

    # 6. Análisis de correlaciones
    correlation_analysis(df)
    print("\nAnálisis de correlaciones completado.")

    # 7. Generar reporte
    generate_summary_report(df, stats_promedios, success_stats)
    print("\nReporte generado exitosamente.")

    print("\nAnálisis exploratorio completado. Revise los resultados en las carpetas:")
    print("- Visualizaciones: results/figures/exploratory/")
    print("- Tablas: results/tables/exploratory/")
    print("- Reporte: results/reporte_exploratorio.txt")

if __name__ == "__main__":
    main()
```

**Archivo: 5_partition.py:**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import json

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
PROCESSED_DIR = DATA_DIR / "processed"
MODELS_DIR = DATA_DIR / "models"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "partition"
TABLES_DIR = RESULTS_DIR / "tables" / "partition"

# Crear directorios necesarios
for dir_path in [MODELS_DIR, FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)
    print(f"Directorio creado/verificado: {dir_path}")
```

```
def load_data():
    """Cargar el dataset codificado"""
    df = pd.read_csv(PROCESSED_DIR / "dataset_codificado.csv")

    # Eliminar columna de código de estudiante si existe
    if 'estudiante.codigo' in df.columns:
        df = df.drop('estudiante.codigo', axis=1)

    print(f"Dataset cargado con {df.shape[0]} observaciones y {df.shape[1]} variables.")
    return df

def prepare_data(df):
    """Preparar los datos para el modelado"""

    # Definir variable objetivo
    target_var = 'egresado.Exito_profesional'

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

    # Separar variable objetivo y predictores
    y = df[target_var]
    X = df.drop(target_var, axis=1)

    print(f"Variable objetivo: {target_var} con {y.nunique()} categorías distintas.")
    print(f"Número de predictores: {X.shape[1]}")

    return X, y

def analyze_class_distribution(y):
    """Analizar la distribución de clases en la variable objetivo"""
    class_counts = y.value_counts()
    print("\nDistribución de clases en la variable objetivo:")
    for clase, count in class_counts.items():
        print(f"Clase {clase}: {count} muestras ({count/len(y):.1%})")
    return class_counts

def partition_data(X, y, train_size=0.7, val_size=0.15, test_size=0.15, random_state=42):
    """Particionar los datos en conjuntos de entrenamiento, validación y prueba"""

    # Verificar que los porcentajes suman 1
    if abs(train_size + val_size + test_size - 1.0) > 0.001:
        raise ValueError("Los porcentajes de partición deben sumar 1.0")

    # Analizar distribución de clases
    class_counts = analyze_class_distribution(y)
    min_samples = class_counts.min()

    # Decidir si usar estratificación
    use_stratify = min_samples >= 2
    stratify_param = y if use_stratify else None
```

```
    if not use_stratify:
        print("\nADVERTENCIA: No se puede usar estratificación debido a clases con
          muy pocas muestras.")
        print("Se realizará una partición aleatoria simple.")

    # Primera división: separar conjunto de entrenamiento
    X_train, X_temp, y_train, y_temp = train_test_split(
        X, y,
        test_size=(val_size + test_size),
        random_state=random_state,
        stratify=stratify_param
    )

    # Segunda división: separar validación y prueba
    test_ratio = test_size / (val_size + test_size)
    X_val, X_test, y_val, y_test = train_test_split(
        X_temp, y_temp,
        test_size=test_ratio,
        random_state=random_state,
        stratify=y_temp if use_stratify else None
    )

    # Verificar y reportar tamaños
    total_samples = len(y)
    print(f"\nPartición de datos completada:")
    print(f"Total de muestras: {total_samples}")
    print(f"Entrenamiento: {len(y_train)} muestras ({len(y_train)/total_samples:.1%})")
    print(f"Validación: {len(y_val)} muestras ({len(y_val)/total_samples:.1%})")
    print(f"Prueba: {len(y_test)} muestras ({len(y_test)/total_samples:.1%})")

    # Verificar distribución de clases en cada conjunto
    print("\nDistribución de clases en cada conjunto:")
    print("\nEntrenamiento:")
    print(y_train.value_counts(normalize=True).round(3))
    print("\nValidación:")
    print(y_val.value_counts(normalize=True).round(3))
    print("\nPrueba:")
    print(y_test.value_counts(normalize=True).round(3))

    return X_train, X_val, X_test, y_train, y_val, y_test

def save_datasets(X_train, X_val, X_test, y_train, y_val, y_test):
    """Guardar los conjuntos de datos particionados"""

    # Crear dataframes completos (X + y)
    train_df = pd.concat([X_train, y_train], axis=1)
    val_df = pd.concat([X_val, y_val], axis=1)
    test_df = pd.concat([X_test, y_test], axis=1)

    # Guardar en formato CSV
    train_df.to_csv(MODELS_DIR / "train_set.csv", index=False)
    val_df.to_csv(MODELS_DIR / "validation_set.csv", index=False)
    test_df.to_csv(MODELS_DIR / "test_set.csv", index=False)

    print("\nConjuntos de datos guardados en la carpeta 'data/models'")
```

```
def analyze_partitions(y_train, y_val, y_test):
    """Analizar la distribución de la variable objetivo en cada partición"""

    # Contar valores en cada conjunto
    train_counts = y_train.value_counts(normalize=True) * 100
    val_counts = y_val.value_counts(normalize=True) * 100
    test_counts = y_test.value_counts(normalize=True) * 100

    # Crear tabla comparativa
    comparison = pd.DataFrame({
        'Entrenamiento (%)': train_counts,
        'Validación (%)': val_counts,
        'Prueba (%)': test_counts
    }).round(1)

    # Guardar tabla
    comparison.to_csv(TABLES_DIR / "distribucion_objetivo_particiones.csv")

    # Crear gráfico de barras agrupadas
    plt.figure(figsize=(12, 6))
    comparison.plot(kind='bar')
    plt.title('Distribución de la Variable Objetivo en cada Partición')
    plt.xlabel('Clase (Éxito Profesional)')
    plt.ylabel('Porcentaje (%)')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "distribucion_particiones.png", bbox_inches='tight')
    plt.close()

    return comparison

def generate_partition_metadata(X_train, X_val, X_test, y_train, y_val, y_test):
    """Generar metadatos sobre la partición"""
    metadata = {
        "total_samples": len(y_train) + len(y_val) + len(y_test),
        "train_samples": len(y_train),
        "validation_samples": len(y_val),
        "test_samples": len(y_test),
        "train_percentage": len(y_train) / (len(y_train) + len(y_val) + len(y_test)),
        "validation_percentage": len(y_val) / (len(y_train) + len(y_val) + len(y_test)),
        "test_percentage": len(y_test) / (len(y_train) + len(y_val) + len(y_test)),
        "predictors_count": X_train.shape[1],
        "class_distribution": {
            "train": y_train.value_counts().to_dict(),
            "validation": y_val.value_counts().to_dict(),
            "test": y_test.value_counts().to_dict()
        },
        "random_state": 42
    }

    # Guardar metadata como JSON
    with open(MODELS_DIR / "partition_metadata.json", 'w') as f:
        json.dump(metadata, f, indent=4)
```

```
    return metadata

    # Guardar reporte de partición
    with open(RESULTS_DIR / "reporte_particion.txt", 'w', encoding='utf-8') as f:
        f.write("REPORTE DE PARTICIÓN DE DATOS\n")
        f.write("============================\n\n")
        f.write(f"Total de observaciones: {metadata['total_samples']}\n\n")
        f.write("Distribución de particiones:\n")
                f.write(f"- Entrenamiento: {metadata['train_samples']} muestras
            ({metadata['train_percentage']:.1%})\n")
                f.write(f"- Validación: {metadata['validation_samples']} muestras
            ({metadata['validation_percentage']:.1%})\n")
                    f.write(f"- Prueba: {metadata['test_samples']} muestras
            ({metadata['test_percentage']:.1%})\n\n")
        f.write(f"Número de predictores: {metadata['predictors_count']}\n")
        f.write("Método de partición: Train-Test-Split con muestreo estratificado\n")
        f.write("Semilla aleatoria: 42\n\n")
        f.write("Archivos generados:\n")
            f.write("- Conjuntos de datos: data/models/train_set.csv, validation_set.csv,
            test_set.csv\n")
                                f.write("- Tablas de distribución:
            results/tables/partition/distribucion_objetivo_particiones.csv\n")
        f.write("- Gráficos: results/figures/partition/distribucion_particiones.png\n")

def main():
    print("Iniciando partición de datos para modelado...")

    # 1. Cargar datos
    df = load_data()

    # 2. Preparar datos
    X, y = prepare_data(df)

    # 3. Particionar datos
    X_train, X_val, X_test, y_train, y_val, y_test = partition_data(X, y)

    # 4. Guardar conjuntos
    save_datasets(X_train, X_val, X_test, y_train, y_val, y_test)

    # 5. Analizar particiones
    comparison = analyze_partitions(y_train, y_val, y_test)
    print("\nDistribución de la variable objetivo en las particiones:")
    print(comparison)

    # 6. Generar metadatos
    generate_partition_metadata(X_train, X_val, X_test, y_train, y_val, y_test)

    print("\nProceso de partición completado exitosamente.")
    print("Revise los reportes en 'results/tables/partition' y 'results/figures/partition'")

if __name__ == "__main__":
    main()
```

**Archivo: 5_partition.py:**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import json

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
PROCESSED_DIR = DATA_DIR / "processed"
MODELS_DIR = DATA_DIR / "models"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "partition"
TABLES_DIR = RESULTS_DIR / "tables" / "partition"

# Crear directorios necesarios
for dir_path in [MODELS_DIR, FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)
    print(f"Directorio creado/verificado: {dir_path}")

def load_data():
    """Cargar el dataset codificado"""
    df = pd.read_csv(PROCESSED_DIR / "dataset_codificado.csv")

    # Eliminar columna de código de estudiante si existe
    if 'estudiante.codigo' in df.columns:
        df = df.drop('estudiante.codigo', axis=1)

     print(f"Dataset cargado con {df.shape[0]} observaciones y {df.shape[1]}
          variables.")
    return df

def prepare_data(df):
    """Preparar los datos para el modelado"""

    # Definir variable objetivo
    target_var = 'egresado.Exito_profesional'

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

    # Separar variable objetivo y predictores
    y = df[target_var]
    X = df.drop(target_var, axis=1)
```

```
        print(f"Variable objetivo: {target_var} con {y.nunique()} categorías distintas.")
    print(f"Número de predictores: {X.shape[1]}")

    return X, y

def analyze_class_distribution(y):
    """Analizar la distribución de clases en la variable objetivo"""
    class_counts = y.value_counts()
    print("\nDistribución de clases en la variable objetivo:")
    for clase, count in class_counts.items():
        print(f"Clase {clase}: {count} muestras ({count/len(y):.1%})")
    return class_counts

def partition_data(X, y, train_size=0.7, val_size=0.15, test_size=0.15, random_state=42):
    """Particionar los datos en conjuntos de entrenamiento, validación y prueba"""

    # Verificar que los porcentajes suman 1
    if abs(train_size + val_size + test_size - 1.0) > 0.001:
        raise ValueError("Los porcentajes de partición deben sumar 1.0")

    # Analizar distribución de clases
    class_counts = analyze_class_distribution(y)
    min_samples = class_counts.min()

    # Decidir si usar estratificación
    use_stratify = min_samples >= 2
    stratify_param = y if use_stratify else None

    if not use_stratify:
        print("\nADVERTENCIA: No se puede usar estratificación debido a clases con muy pocas muestras.")
        print("Se realizará una partición aleatoria simple.")

    # Primera división: separar conjunto de entrenamiento
    X_train, X_temp, y_train, y_temp = train_test_split(
        X, y,
        test_size=(val_size + test_size),
        random_state=random_state,
        stratify=stratify_param
    )

    # Segunda división: separar validación y prueba
    test_ratio = test_size / (val_size + test_size)
    X_val, X_test, y_val, y_test = train_test_split(
        X_temp, y_temp,
        test_size=test_ratio,
        random_state=random_state,
        stratify=y_temp if use_stratify else None
```

```
    )

    # Verificar y reportar tamaños
    total_samples = len(y)
    print(f"\nPartición de datos completada:")
    print(f"Total de muestras: {total_samples}")
                    print(f"Entrenamiento:        {len(y_train)}        muestras
        ({len(y_train)/total_samples:.1%})")
    print(f"Validación: {len(y_val)} muestras ({len(y_val)/total_samples:.1%})")
    print(f"Prueba: {len(y_test)} muestras ({len(y_test)/total_samples:.1%})")

    # Verificar distribución de clases en cada conjunto
    print("\nDistribución de clases en cada conjunto:")
    print("\nEntrenamiento:")
    print(y_train.value_counts(normalize=True).round(3))
    print("\nValidación:")
    print(y_val.value_counts(normalize=True).round(3))
    print("\nPrueba:")
    print(y_test.value_counts(normalize=True).round(3))

    return X_train, X_val, X_test, y_train, y_val, y_test

def save_datasets(X_train, X_val, X_test, y_train, y_val, y_test):
    """Guardar los conjuntos de datos particionados"""

    # Crear dataframes completos (X + y)
    train_df = pd.concat([X_train, y_train], axis=1)
    val_df = pd.concat([X_val, y_val], axis=1)
    test_df = pd.concat([X_test, y_test], axis=1)

    # Guardar en formato CSV
    train_df.to_csv(MODELS_DIR / "train_set.csv", index=False)
    val_df.to_csv(MODELS_DIR / "validation_set.csv", index=False)
    test_df.to_csv(MODELS_DIR / "test_set.csv", index=False)

    print("\nConjuntos de datos guardados en la carpeta 'data/models'")

def analyze_partitions(y_train, y_val, y_test):
    """Analizar la distribución de la variable objetivo en cada partición"""

    # Contar valores en cada conjunto
    train_counts = y_train.value_counts(normalize=True) * 100
    val_counts = y_val.value_counts(normalize=True) * 100
    test_counts = y_test.value_counts(normalize=True) * 100

    # Crear tabla comparativa
    comparison = pd.DataFrame({
        'Entrenamiento (%)': train_counts,
        'Validación (%)': val_counts,
        'Prueba (%)': test_counts
```

```
    }).round(1)

    # Guardar tabla
    comparison.to_csv(TABLES_DIR / "distribucion_objetivo_particiones.csv")

    # Crear gráfico de barras agrupadas
    plt.figure(figsize=(12, 6))
    comparison.plot(kind='bar')
    plt.title('Distribución de la Variable Objetivo en cada Partición')
    plt.xlabel('Clase (Éxito Profesional)')
    plt.ylabel('Porcentaje (%)')
    plt.xticks(rotation=45)
    plt.tight_layout()
            plt.savefig(FIGURES_DIR / "distribucion_particiones.png",
        bbox_inches='tight')
    plt.close()

    return comparison

def generate_partition_metadata(X_train, X_val, X_test, y_train, y_val, y_test):
    """Generar metadatos sobre la partición"""
    metadata = {
        "total_samples": len(y_train) + len(y_val) + len(y_test),
        "train_samples": len(y_train),
        "validation_samples": len(y_val),
        "test_samples": len(y_test),
        "train_percentage": len(y_train) / (len(y_train) + len(y_val) + len(y_test)),
            "validation_percentage": len(y_val) / (len(y_train) + len(y_val) +
        len(y_test)),
        "test_percentage": len(y_test) / (len(y_train) + len(y_val) + len(y_test)),
        "predictors_count": X_train.shape[1],
        "class_distribution": {
            "train": y_train.value_counts().to_dict(),
            "validation": y_val.value_counts().to_dict(),
            "test": y_test.value_counts().to_dict()
        },
        "random_state": 42
    }

    # Guardar metadata como JSON
    with open(MODELS_DIR / "partition_metadata.json", 'w') as f:
        json.dump(metadata, f, indent=4)

    return metadata

    # Guardar reporte de partición
    with open(RESULTS_DIR / "reporte_particion.txt", 'w', encoding='utf-8') as
            f:
        f.write("REPORTE DE PARTICIÓN DE DATOS\n")
        f.write("============================\n\n")
```

```
        f.write(f"Total de observaciones: {metadata['total_samples']}\n\n")
        f.write("Distribución de particiones:\n")
                f.write(f"- Entrenamiento: {metadata['train_samples']} muestras
          ({metadata['train_percentage']:.1%})\n")
              f.write(f"- Validación: {metadata['validation_samples']} muestras
          ({metadata['validation_percentage']:.1%})\n")
                    f.write(f"- Prueba: {metadata['test_samples']} muestras
          ({metadata['test_percentage']:.1%})\n\n")
        f.write(f"Número de predictores: {metadata['predictors_count']}\n")
       f.write("Método de partición: Train-Test-Split con muestreo estratificado\n")
        f.write("Semilla aleatoria: 42\n\n")
        f.write("Archivos generados:\n")
        f.write("- Conjuntos de datos: data/models/train_set.csv, validation_set.csv,
          test_set.csv\n")
                                f.write("- Tablas de distribución:
          results/tables/partition/distribucion_objetivo_particiones.csv\n")
       f.write("- Gráficos: results/figures/partition/distribucion_particiones.png\n")

def main():
    print("Iniciando partición de datos para modelado...")

    # 1. Cargar datos
    df = load_data()

    # 2. Preparar datos
    X, y = prepare_data(df)

    # 3. Particionar datos
    X_train, X_val, X_test, y_train, y_val, y_test = partition_data(X, y)

    # 4. Guardar conjuntos
    save_datasets(X_train, X_val, X_test, y_train, y_val, y_test)

    # 5. Analizar particiones
    comparison = analyze_partitions(y_train, y_val, y_test)
    print("\nDistribución de la variable objetivo en las particiones:")
    print(comparison)

    # 6. Generar metadatos
    generate_partition_metadata(X_train, X_val, X_test, y_train, y_val, y_test)

    print("\nProceso de partición completado exitosamente.")
          print("Revise los reportes en 'results/tables/partition' y
        'results/figures/partition'")

if __name__ == "__main__":
    main()
```

**Archivo: 6_0_modeling.py:**

```
s import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,
        f1_score

from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
import xgboost as xgb

# Importar imblearn para técnicas de oversampling
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline

# Configurar rutas absolutas y directorios para resultados del modelado
BASE_DIR = Path(__file__).resolve().parent.parent
MODELS_DIR = BASE_DIR / "data" / "models"  # Conjuntos particionados se
        guardaron aquí en 5_partition.py
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "modeling"
TABLES_DIR = RESULTS_DIR / "tables" / "modeling"

# Crear directorios si no existen
for dir_path in [FIGURES_DIR, TABLES_DIR]:
   dir_path.mkdir(parents=True, exist_ok=True)
   print(f"Directorio creado/verificado: {dir_path}")

TARGET_COL = 'egresado.Exito_profesional'

def load_datasets():
   """Cargar conjuntos de entrenamiento, validación y prueba desde data/models."""
   train_df = pd.read_csv(MODELS_DIR / "train_set.csv")
   val_df = pd.read_csv(MODELS_DIR / "validation_set.csv")
   test_df = pd.read_csv(MODELS_DIR / "test_set.csv")

   # Separar predictores y variable objetivo
   X_train = train_df.drop(columns=[TARGET_COL])
   y_train = train_df[TARGET_COL]

   X_val = val_df.drop(columns=[TARGET_COL])
   y_val = val_df[TARGET_COL]

   X_test = test_df.drop(columns=[TARGET_COL])
   y_test = test_df[TARGET_COL]

   print(f"Conjunto de entrenamiento: {X_train.shape[0]} muestras y
        {X_train.shape[1]} predictores.")
   print(f"Conjunto de validación: {X_val.shape[0]} muestras.")
   print(f"Conjunto de prueba: {X_test.shape[0]} muestras.")
```

```
    # Analizar distribución de clases
    print("\nDistribución de clases:")
    print(f"Entrenamiento: {y_train.value_counts().to_dict()}")
    print(f"Validación: {y_val.value_counts().to_dict()}")
    print(f"Prueba: {y_test.value_counts().to_dict()}")

    return X_train, y_train, X_val, y_val, X_test, y_test

def apply_oversampling(X_train, y_train):
    """Aplicar técnica de oversampling (SMOTE) para balancear las clases, con manejo
        especial para clases pequeñas."""
    print("\nAplicando oversampling adaptativo...")

    # Mostrar distribución original
    print("Distribución de clases original:")
    class_counts = pd.Series(y_train).value_counts().sort_index()
    print(class_counts)

    # Determinar el tamaño mínimo de clase
    min_class_size = class_counts.min()

    # Para SMOTE, necesitamos al menos k+1 muestras por clase (donde k es el número
        de vecinos)
    # Si hay clases con menos de 6 muestras, ajustamos el número de vecinos
    if min_class_size < 6:
        # Calcular k_neighbors basado en el tamaño mínimo de clase
        # k debe ser menor que el tamaño de la clase más pequeña
        k_neighbors = min(min_class_size - 1, 5)  # Máximo 5, mínimo 1

        if k_neighbors < 1:
            print("⚠ ADVERTENCIA: Hay clases con solo 1 muestra. No se puede aplicar
             SMOTE.")
            print("Se usará RandomOverSampler en su lugar para todas las clases.")

            # Usar RandomOverSampler para clases muy pequeñas
            from imblearn.over_sampling import RandomOverSampler
            ros = RandomOverSampler(random_state=42)
            X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
        else:
            print(f"Ajustando SMOTE para usar {k_neighbors} vecinos debido a clases
             pequeñas.")
            smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
            try:
                X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
            except ValueError as e:
                print(f"Error con SMOTE: {e}")
                print("Recurriendo a RandomOverSampler...")
                from imblearn.over_sampling import RandomOverSampler
                ros = RandomOverSampler(random_state=42)
                X_resampled, y_resampled = ros.fit_resample(X_train, y_train)
    else:
        # Si todas las clases tienen suficientes muestras, usar SMOTE normal
        smote = SMOTE(random_state=42)
        X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

```
    # Mostrar nueva distribución
    print("Distribución de clases después de oversampling:")
    print(pd.Series(y_resampled).value_counts().sort_index())

    print(f"Muestras originales: {len(y_train)} → Muestras después de oversampling:
        {len(y_resampled)}")

    return X_resampled, y_resampled

def train_random_forest(X_train, y_train):
    """Entrenar RandomForestClassifier usando GridSearchCV."""
    print("\nEntrenando Random Forest...")
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [5, 10, None]
    }

    # Usar StratifiedKFold con parámetros ajustados para manejar clases desbalanceadas
    cv = StratifiedKFold(n_splits=min(5, min(y_train.value_counts())), shuffle=True,
        random_state=42)

    rf = RandomForestClassifier(random_state=42, class_weight='balanced')
    grid_rf = GridSearchCV(rf, param_grid, cv=cv, scoring='f1_macro', n_jobs=-1)
    grid_rf.fit(X_train, y_train)
    print(f"Mejores parámetros Random Forest: {grid_rf.best_params_}")
    print(f"Mejor f1_macro en CV: {grid_rf.best_score_:.4f}")
    return grid_rf.best_estimator_, grid_rf.best_score_

def train_xgboost(X_train, y_train):
    """Entrenar XGBoostClassifier usando GridSearchCV con manejo especial para
        etiquetas."""
    print("\nEntrenando XGBoost...")

    # Codificar las etiquetas para asegurar que sean consecutivas (0, 1, 2, ...)
    le = LabelEncoder()
    y_encoded = le.fit_transform(y_train)

    # Obtener número de clases después de codificar
    n_classes = len(le.classes_)
    print(f"Número de clases después de codificación: {n_classes}")
    print(f"Mapeo de clases: {dict(zip(le.classes_, range(n_classes)))}")

    # Usar StratifiedKFold con parámetros ajustados
    cv = StratifiedKFold(n_splits=min(5, min(pd.Series(y_encoded).value_counts())),
                shuffle=True, random_state=42)

    param_grid = {
        'n_estimators': [50, 100],
        'max_depth': [3, 5],
        'learning_rate': [0.01, 0.1]
    }

    # Configurar XGBoost con parámetros para manejar clases desbalanceadas
    xgb_clf = xgb.XGBClassifier(
        random_state=42,
```

```
            eval_metric='mlogloss',
            use_label_encoder=False,
            objective='multi:softprob',
            num_class=n_classes,
            scale_pos_weight=len(y_train)/y_train.value_counts().min()
        )

        try:
            grid_xgb = GridSearchCV(xgb_clf, param_grid, cv=cv, scoring='f1_macro',
                n_jobs=-1, error_score='raise')
            grid_xgb.fit(X_train, y_encoded)

            # Crear un modelo final con los mejores parámetros pero preparado para manejar
                las etiquetas originales
            best_params = grid_xgb.best_params_
            final_model = xgb.XGBClassifier(
                random_state=42,
                eval_metric='mlogloss',
                use_label_encoder=False,
                objective='multi:softprob',
                num_class=n_classes,
                scale_pos_weight=len(y_train)/y_train.value_counts().min(),
                **best_params
            )

            # Entrenar el modelo final con las etiquetas originales
            final_model.fit(X_train, y_train)

            print(f"Mejores parámetros XGBoost: {best_params}")
            print(f"Mejor f1_macro en CV: {grid_xgb.best_score_:.4f}")
            return final_model, grid_xgb.best_score_

        except Exception as e:
            print(f"Error en entrenamiento de XGBoost: {e}")
            print("Saltando modelo XGBoost debido a error.")
            return None, 0.0

    def train_mlp(X_train, y_train):
        """Entrenar MLPClassifier usando GridSearchCV."""
        print("\nEntrenando MLPClassifier...")

        # Usar StratifiedKFold con parámetros ajustados para manejar clases desbalanceadas
        cv = StratifiedKFold(n_splits=min(5, min(y_train.value_counts())), shuffle=True,
            random_state=42)

        param_grid = {
            'hidden_layer_sizes': [(50,), (100,)],
            'activation': ['relu', 'tanh'],
            'alpha': [0.0001, 0.001],
            'max_iter': [300]
        }

        mlp = MLPClassifier(random_state=42, early_stopping=True)
        grid_mlp = GridSearchCV(mlp, param_grid, cv=cv, scoring='f1_macro', n_jobs=-1)
        grid_mlp.fit(X_train, y_train)
```

```
    print(f"Mejores parámetros MLP: {grid_mlp.best_params_}")
    print(f"Mejor f1_macro en CV: {grid_mlp.best_score_:.4f}")
    return grid_mlp.best_estimator_, grid_mlp.best_score_

def evaluate_model(model, X, y, dataset_name=""):
    """Evaluar el modelo y mostrar métricas."""
    y_pred = model.predict(X)
    acc = accuracy_score(y, y_pred)
    f1 = f1_score(y, y_pred, average='macro')
    print(f"\nEvaluación en {dataset_name}:")
    print(f"Accuracy: {acc:.4f}")
    print(f"Macro F1-score: {f1:.4f}")
    print("\nReporte de clasificación:")
    print(classification_report(y, y_pred))
    cm = confusion_matrix(y, y_pred)
    print("Matriz de confusión:")
    print(cm)
    return acc, f1, cm

def plot_confusion_matrix(cm, classes, title, filename):
    """Graficar la matriz de confusión y guardar la figura."""
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=classes, yticklabels=classes)
    plt.title(title)
    plt.ylabel('Actual')
    plt.xlabel('Predicción')
    plt.tight_layout()
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    plt.close()
    print(f"Confusión guardada en: {filename}")

def main():
    # Cargar conjuntos
    X_train, y_train, X_val, y_val, X_test, y_test = load_datasets()

    # Aplicar oversampling al conjunto de entrenamiento
    X_train_resampled, y_train_resampled = apply_oversampling(X_train, y_train)

    # Entrenar cada modelo con los datos balanceados y evaluar en el conjunto de
        validación
    print("\nEntrenando modelos con datos balanceados por oversampling...")
    rf_model,          rf_cv_score          =          train_random_forest(X_train_resampled,
        y_train_resampled)
    xgb_model, xgb_cv_score = train_xgboost(X_train_resampled, y_train_resampled)
    mlp_model, mlp_cv_score = train_mlp(X_train_resampled, y_train_resampled)

    print("\nEvaluando modelos en el conjunto de validación...")
    print("\nRandom Forest:")
    rf_val_acc,    rf_val_f1,    rf_cm    =    evaluate_model(rf_model,    X_val,    y_val,
        dataset_name="Validación")

    # Solo evaluar XGBoost si el entrenamiento fue exitoso
    if xgb_model is not None:
        print("\nXGBoost:")
```

```
    xgb_val_acc, xgb_val_f1, xgb_cm = evaluate_model(xgb_model, X_val, y_val,
        dataset_name="Validación")
else:
    xgb_val_acc, xgb_val_f1, xgb_cm = 0, 0, None
    print("\nXGBoost: Modelo no disponible debido a error en entrenamiento")

print("\nMLPClassifier:")
mlp_val_acc, mlp_val_f1, mlp_cm = evaluate_model(mlp_model, X_val, y_val,
    dataset_name="Validación")

# Seleccionar el mejor modelo basado en macro F1-score en validación
models_scores = {
    "Random Forest": rf_val_f1,
    "XGBoost": xgb_val_f1 if xgb_model is not None else 0,
    "MLP": mlp_val_f1
}
best_model_name = max(models_scores, key=models_scores.get)
print(f"\nEl mejor modelo según el Macro F1-score en validación es:
    {best_model_name}")

if best_model_name == "Random Forest":
    best_model = rf_model
    best_cm = rf_cm
elif best_model_name == "XGBoost" and xgb_model is not None:
    best_model = xgb_model
    best_cm = xgb_cm
else:
    best_model = mlp_model
    best_cm = mlp_cm

# Evaluación final en el conjunto de prueba
print("\nEvaluación final en el conjunto de prueba:")
test_acc, test_f1, test_cm = evaluate_model(best_model, X_test, y_test,
    dataset_name="Prueba")

# Guardar reporte de comparación de modelos
comparison_df = pd.DataFrame({
    "Modelo": ["Random Forest", "XGBoost", "MLPClassifier"],
    "CV F1_macro": [rf_cv_score, xgb_cv_score, mlp_cv_score],
    "Validación F1_macro": [rf_val_f1, xgb_val_f1, mlp_val_f1]
})
comparison_path = TABLES_DIR / "model_comparison.csv"
comparison_df.to_csv(comparison_path, index=False)
print(f"\nReporte de comparación de modelos guardado en: {comparison_path}")

# Guardar la matriz de confusión del mejor modelo en validación y prueba
class_labels = sorted(y_train.unique())
cm_val_filename = FIGURES_DIR / "confusion_matrix_validation.png"
plot_confusion_matrix(rf_cm if best_model_name=="Random Forest" else (xgb_cm
    if best_model_name=="XGBoost" else mlp_cm),
                classes=class_labels,
                title=f"Matriz de Confusión - {best_model_name} (Validación)",
                filename=cm_val_filename)

cm_test_filename = FIGURES_DIR / "confusion_matrix_test.png"
```

```
    plot_confusion_matrix(test_cm, classes=class_labels,
                          title=f"Matriz de Confusión - {best_model_name} (Prueba)",
                          filename=cm_test_filename)

    # Guardar el modelo
    import joblib
    model_path = MODELS_DIR / "best_model.pkl"
    joblib.dump(best_model, model_path)
    print(f"Mejor modelo guardado en: {model_path}")

    # También guardar información de columnas para uso futuro
    feature_names = X_train.columns.tolist()
    with open(MODELS_DIR / "feature_names.pkl", 'wb') as f:
        joblib.dump(feature_names, f)

    print("\nProceso de modelado completado exitosamente.")

if __name__ == "__main__":
    main()
```

**Archivo: 6_3_model_interpretation.py:**

```
s import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import joblib
from sklearn.ensemble import RandomForestClassifier

# Configurar rutas absolutas
BASE_DIR = Path(__file__).resolve().parent.parent
MODELS_DIR = BASE_DIR / "data" / "models"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "modeling"
TABLES_DIR = RESULTS_DIR / "tables" / "modeling"

# Asegurarse que los directorios existan
for dir_path in [FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)

TARGET_COL = 'egresado.Exito_profesional'

def load_datasets():
    """Cargar conjuntos de datos."""
    train_df = pd.read_csv(MODELS_DIR / "train_set.csv")

    # Separar predictores y variable objetivo
    X_train = train_df.drop(columns=[TARGET_COL])
    y_train = train_df[TARGET_COL]

    return X_train, y_train

def train_rf_model(X_train, y_train):
    """Entrenar un modelo Random Forest con los mejores parámetros encontrados."""
    best_params = {'max_depth': 5, 'n_estimators': 50}
```

```
    rf = RandomForestClassifier(random_state=42, class_weight='balanced',
        **best_params)
    rf.fit(X_train, y_train)
    return rf

def plot_feature_importance(model, X_train, top_n=20):
    """Graficar la importancia de las características."""
    # Obtener importancia de características
    feature_importance = model.feature_importances_
    feature_names = X_train.columns

    # Crear DataFrame para mejor manipulación
    fi_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': feature_importance
    }).sort_values(by='Importance', ascending=False)

    # Limitar a las top_n características más importantes
    fi_df_top = fi_df.head(top_n)

    # Graficar
    plt.figure(figsize=(12, 8))
    sns.barplot(x='Importance', y='Feature', data=fi_df_top)
    plt.title(f'Top {top_n} Características Más Importantes (Random Forest)')
    plt.xlabel('Importancia')
    plt.ylabel('Característica')
    plt.tight_layout()
    plt.savefig(FIGURES_DIR / "feature_importance.png", dpi=300,
        bbox_inches='tight')
    plt.close()

    # Guardar tabla completa de importancia
    fi_df.to_csv(TABLES_DIR / "feature_importance.csv", index=False)

    return fi_df

def plot_model_comparison():
    """Graficar comparación de modelos."""
    try:
        # Cargar datos de comparación de modelos
        comparison_df = pd.read_csv(TABLES_DIR / "model_comparison.csv")

        # Convertir datos a formato largo para gráfico
        comparison_long = pd.melt(
            comparison_df,
            id_vars=['Modelo'],
            value_vars=['CV F1_macro', 'Validación F1_macro'],
            var_name='Métrica',
            value_name='F1-score'
        )

        # Graficar
        plt.figure(figsize=(10, 6))
        sns.barplot(x='Modelo', y='F1-score', hue='Métrica', data=comparison_long)
        plt.title('Comparación de Modelos: F1-score')
```

```
        plt.ylim(0, 1.1)  # Escala de 0 a 1.1 para métricas
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.savefig(FIGURES_DIR / "model_comparison.png", dpi=300,
            bbox_inches='tight')
        plt.close()

        print(f"Gráfico de comparación guardado en: {FIGURES_DIR /
            'model_comparison.png'}")

    except Exception as e:
        print(f"Error al crear gráfico de comparación: {e}")

def main():
    print("Generando visualizaciones de interpretación del modelo...")

    # 1. Cargar datos
    X_train, y_train = load_datasets()

    # 2. Entrenar modelo RandomForest (el mejor según resultados previos)
    rf_model = train_rf_model(X_train, y_train)

    # 3. Graficar importancia de características
    fi_df = plot_feature_importance(rf_model, X_train)
    print(f"Top 10 características más importantes:")
    print(fi_df.head(10))
    print(f"Gráfico de importancia guardado en: {FIGURES_DIR /
        'feature_importance.png'}")

    # 4. Graficar comparación de modelos
    plot_model_comparison()

    print("\nProceso completado. Se generaron visualizaciones para interpretación del
        modelo.")

if __name__ == "__main__":
    main()
```

**Archivo: 7_model_application.py:**

```
s import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import sys
import importlib

# CONFIGURACIÓN DE RUTAS
BASE_DIR = Path(__file__).resolve().parent.parent
DATA_DIR = BASE_DIR / "data"
MODELS_DIR = DATA_DIR / "models"
RESULTS_DIR = BASE_DIR / "results"
FIGURES_DIR = RESULTS_DIR / "figures" / "application"
TABLES_DIR = RESULTS_DIR / "tables" / "application"
```

```
# Crear directorios si no existen
for dir_path in [FIGURES_DIR, TABLES_DIR]:
    dir_path.mkdir(parents=True, exist_ok=True)
    print(f"Directorio creado/verificado: {dir_path}")

# Nombre de la variable objetivo
TARGET_COL = 'egresado.Exito_profesional'

def load_or_train_model():
    """Cargar modelo existente o entrenar uno nuevo si no existe"""
    model_path = MODELS_DIR / "best_model.pkl"

    if model_path.exists():
        try:
            best_model = joblib.load(model_path)
            print(f"Modelo cargado desde: {model_path}")
            return best_model
        except Exception as e:
            print(f"Error al cargar el modelo: {e}")

    print("No se encontró un modelo guardado. Entrenando un nuevo modelo...")

    # Importar funciones usando importlib o cargando directamente las funciones
    sys.path.append(str(BASE_DIR))
    try:
        # Método manual para importar desde un archivo con número
        model_interpretation_path = BASE_DIR / "src" / "6_3_model_interpretation.py"

        # Verificar que el archivo existe
        if not model_interpretation_path.exists():
            raise FileNotFoundError(f"No se encontró el archivo
             {model_interpretation_path}")

        # Cargar código manualmente
        import importlib.util
        spec = importlib.util.spec_from_file_location("model_interpretation",
            model_interpretation_path)
        model_interpretation = importlib.util.module_from_spec(spec)
        spec.loader.exec_module(model_interpretation)

        # Ahora podemos acceder a las funciones
        X_train, y_train = model_interpretation.load_datasets()
        model = model_interpretation.train_rf_model(X_train, y_train)

        # Guardar el modelo para uso futuro
        joblib.dump(model, model_path)
        print(f"Nuevo modelo entrenado y guardado en: {model_path}")

        return model
    except Exception as e:
        print(f"Error al entrenar nuevo modelo: {e}")
        print(f"Detalles del error: {str(e)}")
        print("Por favor, ejecute primero 6_0_modeling.py para generar el modelo.")
        sys.exit(1)
```

```
def load_feature_names():
    """Cargar los nombres de características utilizados por el modelo"""
    try:
        feature_path = MODELS_DIR / "feature_names.pkl"
        if feature_path.exists():
            return joblib.load(feature_path)

        # Si no existe, intentar cargar del conjunto de entrenamiento
        train_df = pd.read_csv(MODELS_DIR / "train_set.csv")
        return [col for col in train_df.columns if col != TARGET_COL]

    except Exception as e:
        print(f"Error al cargar nombres de características: {e}")
        return None

def generate_synthetic_data(n_samples=10, features=None):
    """Genera datos sintéticos optimizando la creación del DataFrame"""
    if features is None:
        print("ADVERTENCIA: No se proporcionó lista de características. Usando
            conjunto predefinido.")
        # Cargar lista de características del conjunto de entrenamiento
        train_df = pd.read_csv(MODELS_DIR / "train_set.csv")
        features = [col for col in train_df.columns if col != TARGET_COL]

    print(f"Generando datos sintéticos con {len(features)} características")

    # Crear diccionario de datos
    data = {}
    for feature in features:
        data[feature] = np.zeros(n_samples)

    # Variables numéricas específicas
    if 'estudiante.promedio_nota' in features:
        data['estudiante.promedio_nota'] = np.random.normal(14.12, 1.63,
            n_samples).clip(12.16, 19.41).round(2)

    if 'estudiante.max_ciclo' in features:
        data['estudiante.max_ciclo'] = 10  # Valor constante observado

    if 'egresado.Satisfaccion_actual' in features:
        data['egresado.Satisfaccion_actual'] = np.random.uniform(3, 5,
            n_samples).round(1)

    if 'egresado.Meses_para_primer_empleo' in features:
        data['egresado.Meses_para_primer_empleo'] = np.random.randint(0, 24,
            n_samples)

    # Para columnas dummy (One-Hot Encoding)
    # Establecer un valor 1 en una variable categórica de cada grupo
    dummy_prefixes = [
        'estudiante.DependenciaEconomica_',
        'estudiante.TipoConvivencia_',
        'egresado.Tipo_organizacion_',
        'egresado.Area_desempeno_',
        'egresado.Idiomas_'
```

```
    ]

    for prefix in dummy_prefixes:
        cols = [col for col in features if col.startswith(prefix)]
        if cols:
            for i in range(n_samples):
                col_to_activate = np.random.choice(cols)
                data[col_to_activate][i] = 1

    # Crear DataFrame de una sola vez (evita fragmentación)
    synthetic_df = pd.DataFrame(data)

    return synthetic_df

def main():
    print("Aplicación del modelo de predicción de éxito profesional\n")

    # 1. Cargar el modelo
    best_model = load_or_train_model()

    # 2. Obtener lista de características requeridas
    features = load_feature_names()

    # 3. Generar datos sintéticos
    synthetic_data = generate_synthetic_data(n_samples=10, features=features)
    print("\nMuestra de datos sintéticos generados:")
    print(synthetic_data.head(3))

    # 4. Aplicar el modelo para predecir
    try:
        predictions = best_model.predict(synthetic_data)
        probabilities = best_model.predict_proba(synthetic_data)

        # Agregar predicciones al dataframe
        synthetic_data[TARGET_COL] = predictions

        # Agregar probabilidades de cada clase
        class_labels = best_model.classes_
        for i, label in enumerate(class_labels):
            synthetic_data[f'Prob_{label}'] = probabilities[:, i].round(3)

        print("\nPredicciones del modelo:")
        print(synthetic_data[[TARGET_COL] + [f'Prob_{label}' for label in
            class_labels]].head(10))

        # 5. Guardar resultados
        synthetic_data.to_csv(TABLES_DIR / "synthetic_predictions.csv", index=False)
        print(f"\nPredicciones guardadas en: {TABLES_DIR /
            'synthetic_predictions.csv'}")

        # 6. Visualizaciones
        # Distribución de predicciones
        plt.figure(figsize=(10, 6))
        ax = sns.countplot(x=synthetic_data[TARGET_COL])
        plt.title("Distribución de Éxito Profesional Predicho")
```

```
        plt.xlabel("Nivel de Éxito Profesional")
        plt.ylabel("Cantidad de Egresados")

        # Agregar etiquetas con porcentajes
        total = len(synthetic_data)
        for p in ax.patches:
            percentage = '{:.1f}%'.format(100 * p.get_height()/total)
            x = p.get_x() + p.get_width()/2
            y = p.get_height()
            ax.annotate(percentage, (x, y), ha='center')

        plt.tight_layout()
        plt.savefig(FIGURES_DIR / "synthetic_predictions_distribution.png", dpi=300,
            bbox_inches='tight')
        plt.close()

        print(f"Gráfico de distribución guardado en: {FIGURES_DIR /
            'synthetic_predictions_distribution.png'}")

    except Exception as e:
        print(f"Error al realizar predicciones: {e}")
        print("Asegúrese de que los datos sintéticos coinciden con el formato esperado por
            el modelo.")

    print("\nProceso de aplicación del modelo completado.")

if __name__ == "__main__":
    main()
```

## ANEXO 3: Diagrama de componentes

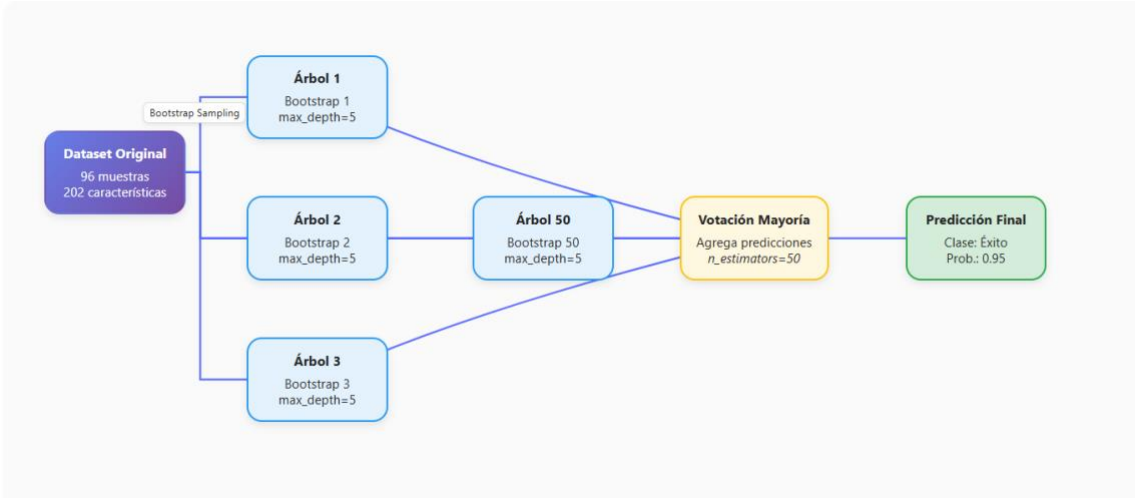


*Figura 34 - Diagrama de Componentes - Random Forest*

**Figura A.1**: Arquitectura del bosque aleatorio implementado. El diagrama presenta el proceso de ensemble learning donde el dataset original (96 muestras, 202 características) se divide mediante bootstrap sampling en 50 subconjuntos independientes. Cada subconjunto entrena un árbol de decisión con profundidad máxima de 5. Las predicciones individuales de los 50 árboles se agregan mediante votación por mayoría para determinar la clase final de éxito profesional con su probabilidad asociada.

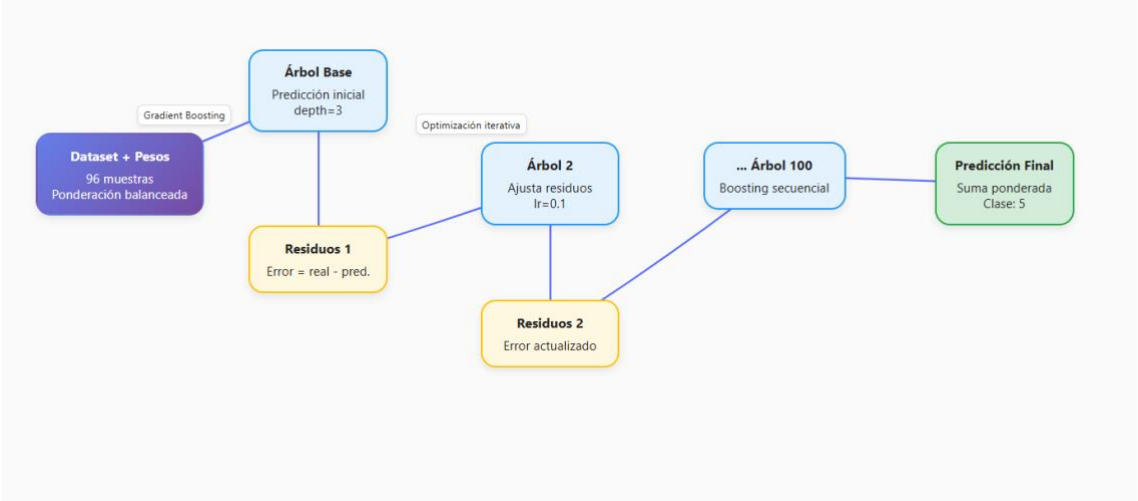


*Figura 35 - Diagrama de Componentes - XGBoost*

**Figura A.2**: Flujo de procesamiento del algoritmo XGBoost. El diagrama ilustra el proceso iterativo de gradient boosting, iniciando con un árbol base de profundidad 3 que genera predicciones iniciales. Cada iteración calcula los residuos (diferencia entre valores reales y predichos) y entrena un nuevo árbol para corregir estos errores con una tasa de aprendizaje de 0.1. El proceso continúa secuencialmente hasta alcanzar 100 árboles, donde la predicción final resulta de la suma ponderada de todos los árboles entrenados.

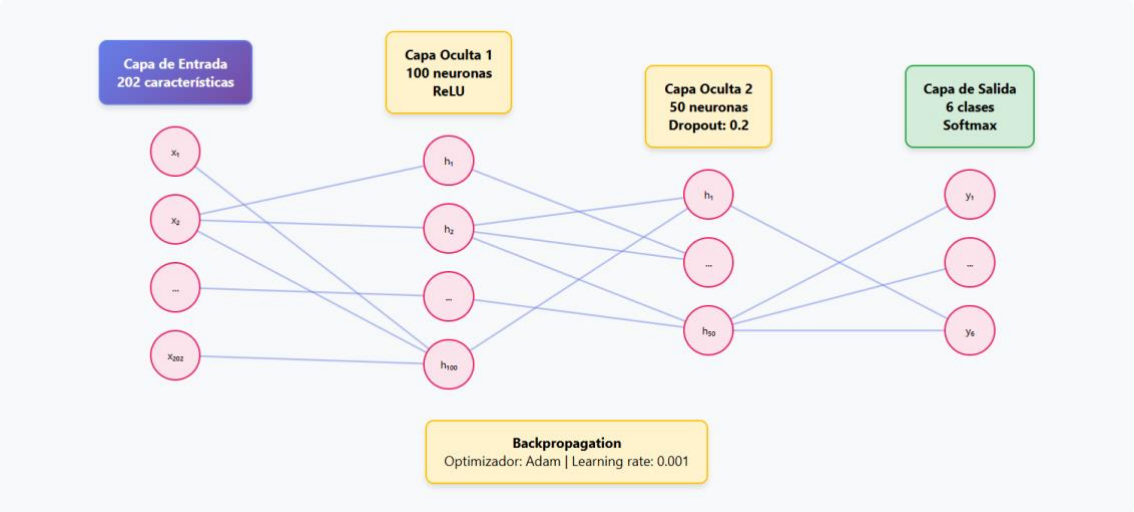


*Figura 36 - Diagrama de Componentes - MLPClassifier*

**Figura A.3**: Arquitectura del perceptrón multicapa implementado. El diagrama muestra la estructura de la red neuronal con 202 neuronas de entrada correspondientes a las características del dataset codificado, dos capas ocultas (100 y 50 neuronas respectivamente) con función de activación ReLU y dropout de 0.2 para prevenir sobreajuste, y una capa de salida con 6 neuronas que representan las clases de éxito profesional mediante función softmax. El proceso de aprendizaje se realiza mediante backpropagation con optimizador Adam y tasa de aprendizaje de 0.001.