UNIVERSIDAD NACIONAL DE MISIONES
+INGENIO Revista de Ciencia Tecnología e Innovación
FACULTAD DE INGENIERÍA UNaM
Vol. 6 N° 2, julio-diciembre 2025, pag. 32-60
ISSN 2618-5520 on line
DOI - https://doi.org/10.36995/j.masingenio.2025.06.02.003

Predicción de Costos y Plazos en Proyectos de Construcción: Un enfoque Basado en Inteligencia Artificial

Miryan R. Puchini [1, *], Nancy B. Ganz [1, 2], Yolanda E. Zado [1]

[1] *Universidad Nacional de Misiones, Facultad de Ingeniería, Misiones, Argentina.*

[2] *Universidad Nacional de Misiones, Facultad de Ciencias Exactas Químicas y Naturales, Instituto de Materiales de Misiones, Misiones, Argentina.*

e-mails: miryanpuchini@fio.unam.edu.ar, nancy.ganz@fio.unam.edu.ar, yolandazado@gmail.com

Resumen

La estimación precisa de tiempo y costos es fundamental para la planificación y gestión eficiente de los proyectos de construcción, ya que la incertidumbre o información imprecisa puede generar sobrecostos, demoras y un uso ineficiente de los recursos. Los métodos tradicional es presentan limitaciones en contextos dinámicos y complejos. En este marco, estudios recientes demuestran que la aplicación de modelos de Inteligencia Artificial y aprendizaje automático mejora significativamente la precisión predictiva al manejar la variabilidad de los datos. Este trabajo se basa en una revisión bibliográfica crítica y propone el uso de modelos de IA como alternativa a los enfoques tradicionales para optimizar la estimación de tiempos y costos. Los resultados indican que modelos como Extra Trees y XGBoost alcanzan coeficientes de determinación (R^2) superiores a 0,92 en la predicción de costos, mientras que la estimación de tiempos depende fuertemente de la calidad de los datos disponibles. Se concluye que la IA constituye una herramienta estratégica para mejorar la planificación y la toma de decisiones en el sector.

Palabras Clave *– Proyectos de construcción, Estimación de costos, Predicción de duración, Inteligencia artificial, Aprendizaje automático.*

Abstract

Accurate time and cost estimation is fundamental for the efficient planning and management of construction projects, as uncertainty or imprecise information can lead to cost overruns, delays, and inefficient resource allocation. Traditional methods exhibit limitations in dynamic and complex contexts. Within this framework, recent studies demonstrate that the application of Artificial Intelligence (AI) and Machine Learning models significantly improves predictive accuracy by handling data variability. This paper is based on a critical literature review and proposes the use of AI models as an alternative to traditional approaches to optimize time and cost estimation. The results indicate that models such as Extra Trees and XGBoost achieve coefficients of determination (R^2) exceeding 0.92 in cost prediction, while time estimation remains heavily dependent on the qudity of available data. It is concluded that AI constitutes a strategic tool for enhancing planning and decision-making within the sector.

Keywords *– Construction projects, Cost estimation, Duration prediction, Artificial intelligence, Machine learning.*

1. Introducción

La precisión en la estimación de tiempo y costos constituye un factor crítico para la planificación y la gestión eficiente de los proyectos de construcción. La presencia de incertidumbres o de información imprecisa en estas etapas puede derivar en sobrecos tos, demoras, desvíos en los cronogramas y un uso ineficiente de los recursos, afectando negativamente a todas las partes involucradas.

*Autor en correspondencia: miryanpuchini@fio.unam.edu.ar / / Revista +Ingenio: masingenio@fio.una.edu.ar

Tradicionalmente, estas estimaciones se han realizado mediante métodos descriptivos, conceptuales o estadísticos, los cuales presentan limitaciones en entornos altamente dinámicos y complejos. En este contexto, investigaciones recientes han demostrado que la incorporación de modelos de Inteligencia Artificial (IA) y aprendizaje automático (*Machine Learning*, ML por sus siglas en inglés) mejora significativamente la precisión predictiva, al emplear técnicas avanzadas como *Random Forest*, K- *Nearest Neighbors*, SVM o redes neuronales, logrando resultados más sólidos frente a la variabilidad de los datos.

Este trabajo se fundamenta en una revisión bibliográfica crítica y propone el desarrollo de modelos basados en IA como alternativa eficaz frente a los métodos tradicionales, con el objetivo de optimizar la estimación de tiempo y costos en proyectos de construcción, reduciendo la incertidumbre y fortaleciendo la toma de decisiones desde etapas tempranas.

2. Marco teórico

2.1. Factores e incertidumbres en los proyectos de construcción

La finalización en tiempo y forma de los proyectos de construcción depende en gran medida de estimaciones precisas de tiempo y costo, dos parámetros que se ven fuertemente afectados por incertidumbres inherentes al entorno constructivo. Estas incertidumbres, si no se gestionan adecuadamente, impactan negativamente en la planificación, ejecución y control del proyecto, por lo que es crucial desarrollar estrategias eficaces para su identificación y mitigación.

En el trabajo consultado han subrayado la importancia de incorporar la incertidumbre en los modelos de programación de obras, mediante el uso de herramientas avanzadas como algoritmos de IA y técnicas estadísticas. Se destaca, además, la influencia de factores como la gestión deficiente del sitio, la escasez de recursos financieros y las fallas comunicacionales, los cuales contribuyen a desviaciones significativas en los cronogramas y presupuestos.

Han abordado estas problemáticas a través de modelos matemáticos capaces de cuantificar el impacto de los riesgos sobre la duración y los costos. El tratamiento explícito de distintos tipos de incertidumbre, ya sea a través de estudios empíricos, curvas S de riesgo o modelos predictivos, han demostrado mejorar considerablemente la precisión de las estimaciones.

Asimismo, evidencian que el sistema de entrega del proyecto influye directamente en la estimación del tiempo, siendo objeto de análisis comparativos, evaluaciones de riesgo y estudios de caso, especialmente en proyectos de vivienda pública. La adopción de enfoques como el diseño-construcción o la entrega integrada de proyectos ha mostrado ventajas frente a contextos de alta complejidad e incertidumbre, al facilitar una mejor coordinación entre las partes y una mayor adaptabilidad frente a eventos no previstos.

Finalmente mencionan, sobre nuevas líneas de investigación que abordan temas emergentes que afectan la entrega de proyectos, como la ciberseguridad en entornos digitales de construcción o la incorporación de indicadores de sostenibilidad en los procesos de planificación y ejecución, ampliando así la visión tradicional de riesgos hacia un enfoque más integral y actual [1].

Respecto a los factores que inciden en los resultados del proyecto, la literatura los clasifica en dos categorías: Factores Cruciales, aquellos que ayudan a prevenir disputas y retrasos, por ejemplo, la estimación precisa de la duración de cada actividad. Factores Críticos de Éxito (CSF, por sus siglas en inglés), que garantizan el éxito en la planificación y permiten una gestión eficiente de los recursos. Algunos CSF identificados son: tamaño y complejidad del proyecto, condiciones del

sitio, tipo de obra, ubicación geográfica, disponibilidad de recursos y materiales, productividad de la mano de obra, modificaciones en el diseño, cambios normativos y condiciones climáticas imprevistas.

Se han identificado hasta 48 factores causales de retrasos y 38 factores de sobrecostos, clasificados en cuatro categorías: causas atribuibles al contratista, al consultor, al cliente y causas externas. Estos estudios constituyen un marco valioso, pero es necesario profundizar en su análisis para desarrollar soluciones viables y evaluar su eficacia en la práctica, con el objetivo de asegurar la ejecución exitosa de los proyectos de construcción [2].

2.2. Métodos tradicionales para estimar tiempo y costo

Los proyectos de construcción, especialmente los del tipo residencial, requieren estimaciones precisas de la duración de cada actividad debido a los diversos riesgos e incertidumbres que afectan el tiempo total de finalización. Tradicionalmente, estas estimaciones se realizan mediante técnicas clásicas de planificación y gestión de proyectos, entre las que se destacan:

- El Método del Camino Crítico (CPM, por sus siglas en inglés *Critical Path Method*) es un enfoque determinista ampliamente utilizado en la planificación de proyectos de construcción. Su principal ventaja radica en la simplicidad de su implementación, ya que permite identificar la duración mínima del proyecto a partir de un análisis lineal. Sin embargo, una de sus limitaciones más significativas es que considera una única duración fija para cada actividad, sin incorporar la variabilidad ni los factores de incertidumbre. Como resultado, el cronograma generado puede no reflejar fielmente las condiciones reales del proyecto [3].
- La Técnica de Evaluación y Revisión de Programas (PERT, por sus siglas en inglés *Program Evaluation and Review Technique*), introduce un enfoque probabilístico al considerar tres estimaciones de duración (optimista, más probable y pesimista) para cada actividad. Este método permite incorporar la incertidumbre inherente a los proyectos, generando una planificación más realista y detallada, al tiempo que facilita la visualización de tareas y sus respectivas dependencias [4].
- El Método de Simulación de Montecarlo (MCS) es una técnica estadística avanzada que permite simular múltiples escenarios posibles en función de variables aleatorias. Su aplicación no se limita solo a la estimación de tiempos, sino que también considera restricciones de costo y riesgos asociados, lo cual proporciona un marco más robusto para la toma de decisiones estratégicas durante la planificación y ejecución del proyecto [5].

Además, en el estudio [5] se ejemplifica la aplicabilidad de estos métodos tradicionales en un proyecto de construcción de una vivienda con 24 actividades planificadas. Al aplicar CPM, se obtuvo una duración estimada de 63 días. En contraste, al emplear PERT y MCS con un nivel de confianza del 95 %, las duraciones proyectadas fueron de 70 y 79 días, respectivamente. Estos resultados evidencian que los métodos probabilísticos, como PERT y MCS, ofrecen cronogramas más ajustados a la realidad al contemplar los riesgos e incertidumbres propios de los proyectos de construcción.

Estos resultados demuestran que los métodos probabilísticos constituyen herramientas valiosas para los gerentes de proyectos, permitiéndoles programar obras de construcción residencial

considerando los riesgos e incertidumbres inherentes. Si bien PERT y MCS son útilespara la elaboración de cronogramas más realistas, su precisión está directamente condicionada por la calidad de las variables de entrada y por la fidelidad del modelo que representa al proyecto. Las deficiencias de estos modelos, están en que las estimaciones de duración de las actividades no son precisas, perdiendo confiabilidad en sus resultados. Es importante destacar que ni PERT, ni MCS proporcionan soluciones definitivas por sí mismos, sino que actúan como herramientas de apoyo para anticipar el comportamiento del proyecto en escenarios inciertos y la toma de decisiones finales recae en el gerente de proyecto, quien debe realizar una evaluación integral de las condiciones particulares de la obra antes de adoptar una estrategia de planificación adecuada [6].

2. 3. Modelación de Información de la Construcción

Como se ha mencionado previamente, los métodos tradicionales, deterministas o probabilísticos, utilizados en la planificación de proyectos de construcción presentan limitaciones significativas, especialmente en lo que respecta a la integración de información multidimensional y la actualización en tiempo real.

En respuesta a estas limitaciones, el Modelado de Información de la Construcción (*Building Information Modeling*, BIM) se posiciona como una tecnología disruptiva que ha transformado la forma en que se planifican, diseñan, ejecutan y gestionan los proyectos. BIM opera dentro de un entorno colaborativo e integrado, facilitando la centralización de datos y mejorando la coordinación entre los distintos actores del proceso constructivo.

En la actualidad, numerosos profesionales del sector de la arquitectura, la ingeniería y la construcción (AEC) reconocen un cambio de paradigma marcado por la convergencia entre BIM y la IA. Esta integración promueve nuevas oportunidades para optimizar el rendimiento organizacional, mejorar la toma de decisiones y aumentar la eficiencia en la entrega de proyectos, especialmente mediante el aprovechamiento de grandes volúmenes de datos secundarios generados durante el ciclo de vida del proyecto.

BIM puede entenderse a partir de sus dos componentes fundamentales: por un lado, como una representación digital compartida de un activo físico, que constituye una base confiable para la toma de decisiones a lo largo de todo su ciclo de vida, desde su concepción hasta la demolición; y por otro lado, como un proceso basado en modelos 3D inteligentes, que proporciona a los profesionales del sector AEC las herramientas y la información necesarias para planificar, diseñar, construir y operar edificaciones e infraestructuras de manera más eficiente.

Un concepto clave de BIM es su estructura en una serie de capas interconectadas, cada una de las cuales aporta niveles adicionales de información y funcionalidad al modelo, enriqueciendo su capacidad para representar y gestionar el ciclo de vida de un proyecto de construcción. Comienza con la capa geométrica (3D), que representa la forma y ubicación de los elementos del edificio, y se complementa con la capa semántica, que agrega propiedades y atributos técnicos. La capa topológica define relaciones funcionales entre componentes, mientras que las capas temporales (4D) y costos (5D) incorporan planificación y estimación financiera. A esto se suman la capa de sostenibilidad (6D), que evalúa el impacto ambiental, la gestión de instalaciones (7D), que abarca el mantenimiento y uso eficiente del edificio, y finalmente la integración con IoT, que conecta sensores y dispositivos inteligentes para monitoreo en tiempo real.

2.3.1. Rol de la IA en BIM

La integración entre la IA y BIM está dando lugar a modelos inteligentes capaces de procesar grandes volúmenes de datos en tiempo real, reconocer patrones y proponer soluciones optimizadas en todo el ciclo de vida del proyecto. Esta sinergia ha impulsado el desarrollo de múltiples aplicaciones especializadas, acompañadas de herramientas concretas en el mercado:

• Diseño paramétrico y generativo: Autodesk Forma, permite generar múltiples opciones de diseño en función de datos del sitio y normativas; Project Refinery, aplica IA para explorar alternativas según objetivos específicos.

• Análisis energético y sostenibilidad: Autodesk Insight, integrado con Revit, analiza el rendimiento energético del edificio; Tally, complemento de Revit, calcula el impacto ambiental de los materiales.

• Detección de conflictos y coordinación: Navisworks, con capacidades de IA, identifica conflictos y sugiere resoluciones basadas en datos históricos.

• Simulación de construcción y planificación (4D): Synchro, permite simular escenarios, optimizar cronogramas y prever retrasos.

• Cumplimiento normativo automatizado: UpCodes AI, vinculado con Revit, verifica automáticamente la conformidad con códigos de edificación.

• Evaluación de riesgos del proyecto: Kognoz, aplica ML para anticipar riesgos combinando datos BIM e históricos.

• Gestión inteligente de instalaciones (7D): IBM TRIRIGA, integrado con BIM, optimiza el mantenimiento, la gestión de activos y el uso eficiente del espacio.

• Modificaciones de diseño en tiempo real: Revit con IA, puede sugerir cambios basados en costos, rendimiento y regulaciones.

La fusión entre IA y BIM no solo mejora la eficiencia operativa, sino que redefine la forma en que se conciben, construyen y mantienen los entornos construidos. Las aplicaciones actuales ya impactan en áreas clave como diseño generativo, análisis de sostenibilidad, planificación 4D, gestión de riesgos y mantenimiento predictivo.

Las investigaciones apuntan al desarrollo de algoritmos de IA más robustos, la incorporación de computación cuántica, técnicas de aprendizaje federado, IA explicable, biomimética, blockchain, y gemelos digitales inteligentes aplicados a ciudades. Este ecosistema tecnológico promete transformar radicalmente la industria de la AEC, facilitando la creación de entornos más sostenibles, resilientes e inteligentes [7].

De lo expuesto se desprende que la implementación efectiva de la metodología BIM, especialmente en combinación con herramientas de IA, requiere la utilización de múltiples softwares especializados que abordan distintas dimensiones del proyecto (diseño, análisis energético, planificación, gestión de riesgos, mantenimiento, entre otros). Esta multiplicidad de plataformas, muchas veces desarrolladas por distintos proveedores y con niveles variables de interoperabilidad, genera una fragmentación de la información. Como resultado, no es posible disponer de una base de datos única y centralizada que contenga de forma integrada todos los datos

del proyecto. Esta dispersión dificulta la trazabilidad completa, la sincronización entre disciplinas y la aplicación de modelos de análisis globales basados en datos unificados.

2.4. Automatizar las estimaciones del tiempo y costo de un proyecto de construcción

En disciplinas ajenas a las Cienciasde la Computación, los términos Inteligencia Artificial (IA), aprendizaje automático o ML, y aprendizaje profundo o Deep Learning (DL, por sus siglas en inglés) suelen utilizarse de manera indistinta, especialmente en contextos comerciales. Sin embargo, desde una perspectiva técnica y académica, presentan diferencias sustanciales que es importante esclarecer (ver Fig. 1).

La IA abarca el conjunto de técnicas que permiten a las máquinas imitar capacidades cognitivas humanas, como el razonamiento, la toma de decisiones, el reconocimiento de patrones o el procesamiento del lenguaje. El ML constituye un subconjunto de la IA, que se enfoca en desarrollar algoritmos capaces de aprender a partir de datos, identificar patrones y realizar predicciones sin ser explícitamente programados para cada tarea específica. Algunos algoritmos de ML utilizados para estimaciones incluyen la regresión lineal, los árboles de decisión y los métodos basados en vecinos más cercanos (K-*Nearest Neighbors* por sus siglas en inglés KNN). El aprendizaje profundo es un subconjunto dentro del ML que se basa en arquitecturas de redes neuronales artificiales compuestas por múltiples capas de procesamiento. Estas redes permiten modelar relaciones no lineales complejas y extraer representaciones jerárquicas de los datos, mejorando el rendimiento en tareas como el reconocimiento de imágenes, el procesamiento de lenguaje natural y la predicción de variables en contextos de alta dimensionalidad, siendo eficaz en contextos donde los datos son masivos y no estructurados [8].

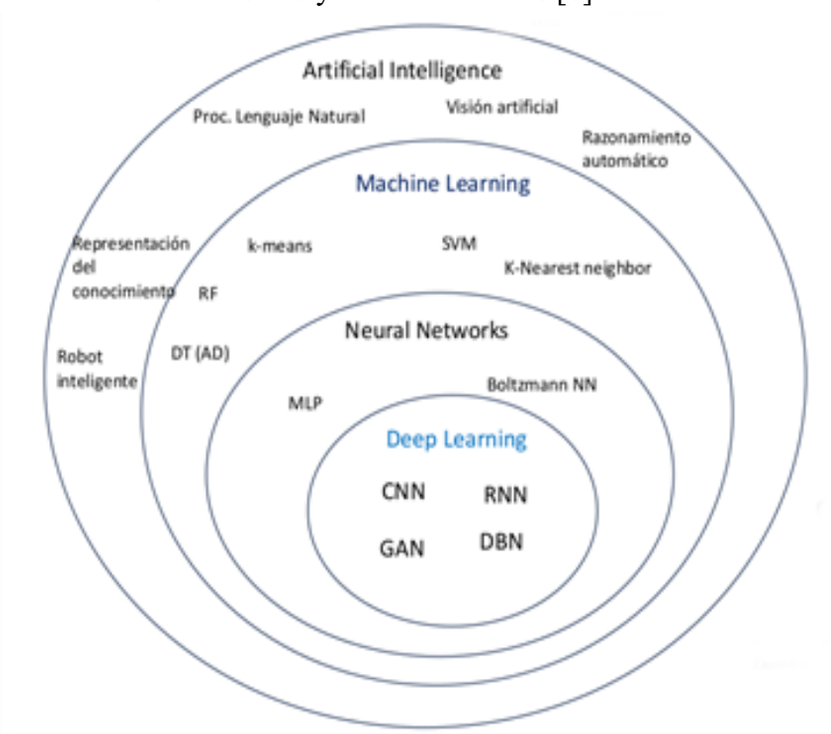


Fig. 1. Taxonomía IA [9].

Diversas investigaciones han evidenciado que la integración de modelos de inteligencia artificial con técnicas estadísticas mejora significativamente la precisión en la estimación de costos y plazos en proyectos de construcción. Se han aplicado algoritmos de aprendizaje automático como

Máquinas de Vectores de Soporte (SVM), Bosques Aleatorios, Naive Bayes y Redes Neuronales Artificiales (ANN), superando las limitaciones de métodos tradicionales como PERT, CPM y simulación de Montecarlo.

Entre los enfoques más prometedores destacan los modelos híbridos, que combinan distintos algoritmos para aprovechar sus fortalezas individuales, logrando mejoras sustanciales en la estimación, con errores MAPE (por sus siglas en inglés *Mean Absolute Percentage Error*) reportados inferiores al 5 % en costos y al 8 % en tiempo. Además, se ha incorporado la incertidumbre como variable crítica, considerando tanto componentes aleatorios como epistémicos, lo que permite una planificación más robusta y adaptativa.

En el ámbito del aprendizaje profundo, las Redes Neuronales Convolucionales (CNN) han mostrado potencial para estimar duraciones a partir de datos visuales o topográficos, mientras que las Redes Neuronales Recurrentes (RNN) destacan por su capacidad para modelar dependencias temporales y patrones secuenciales en actividades de obra. Estos avances posicionan a las arquitecturas neuronales, especialmente en configuraciones híbridas, como herramienta clave en la transformación digital del sector AEC, permitiendo anticipar desviaciones, reducir la incertidumbre y optimizar la planificación desde etapas tempranas del proyecto [10].

Una investigación reciente abordó la comparación entre algoritmos de aprendizaje automático y métodos tradicionales para predecir la duración de proyectos de construcción, utilizando datos recolectados mediante encuestas a organizaciones del sector. Se evaluaron modelos como Redes Neuronales Artificiales (NN), *Random Forest* (RF), Máquinas de Vectores de Soporte (SVM), K-*Nearest Neighbors* (KNN) y árboles de regresión (CART), implementados en R con la librería caret.

Los resultados indicaron que KNN fue el modelo más preciso, con un RMSE (por sus siglas en inglés Root Mean Square Error) de 81 días y un R^2 de 0,97, seguido de cerca por RF. En contraste, las redes neuronales mostraron bajo rendimiento debido a limitaciones en el tamaño del conjunto de datos y la selección de variables. El estudio concluye que RF y KNN ofrecen alta precisión para proyectos públicos, pero destaca la necesidad de incorporar más datos y variables contextuales para mejorar la robustez de modelos más complejos, como los de aprendizaje profundo [11].

2.5. Modelos predictivos basados en arboles de decisión

Si bien el avance de la IA y del ML ha impulsado el desarrollo de modelos predictivos de alta precisión aplicables a múltiples dominios, incluyendo la industria de la construcción; los algoritmos basados en árboles de decisión constituyen una de las familias más utilizadas por su capacidad para modelar relaciones no lineales y manejar variables heterogéneas. Entre ellos, los modelos *Random Forest* y *Extremely Randomized Trees* (ExtraTrees) se destacan por su robustez ante ruido y su estabilidad frente al sobreajuste, al combinar múltiples árboles entrenados sobre subconjuntos aleatorios de los datos. [12]. Estudios recientes evidencian que las variantes modernas de estos algoritmos, como SPLDExtraTrees, logran mejorar la generalización mediante mayor aleatorización en los puntos de corte. Asimismo, investigaciones comparativas demuestran que los métodos de ensamble basados en árboles —como *Random Forest*, ExtraTrees y BART— presentan un excelente equilibrio entre precisión y explicabilidad en entornos con alta incertidumbre [13].

Dentro de los modelos de *boosting*, XGBoost y LightGBM son ampliamente reconocidos por su capacidad para optimizar tanto el rendimiento como la eficiencia computacional. XGBoost

emplea técnicas de regularización y poda que reducen el error de generalización, mientras que LightGBM introduce métodos edmuestreo y compresión de histogramas que aceleran el entrenamiento sin pérdida significativa de precisión. Aplicaciones recientes confirman la efectividad de XGBoost en la predicción de variables económicas y de ingeniería, superando a modelos lineales tradicionales. En paralelo, la literatura también destaca el uso de descenso estocástico del gradiente (SGD) como método de optimización eficaz para modelos lineales y redes neuronales en contextos de grandes volúmenes de datos [14].

Más recientemente, las herramientas de AutoML han permitido automatizar el proceso de selección de modelos y ajuste de hiperparámetros. En particular, FLAML (*Fast Lightweight AutoML*) propone un enfoque eficiente y de bajo costo computacional que prioriza la precisión en escenarios con recursos limitados, mostrando resultados competitivos frente a pipelines manuales. En el ámbito específico de la construcción, diversos estudios de revisión confirman que los algoritmos de aprendizaje automático, en especial los modelos de ensamble, han sido los más utilizados para predecir costos, duraciones y productividad de obra, destacando su potencial para reducir la incertidumbre y optimizar la toma de decisiones [15].

3. Materiales y métodos

Este estudio adopta un enfoque basado en el flujo de trabajo típico del aprendizaje automático (ML), ampliamente utilizado por empresas tecnológicas líderes como Amazon, Google y Microsoft. El proceso se estructura en tres etapas principales: tratamiento de los datos, modelado e implementación (ver Fig. 2).

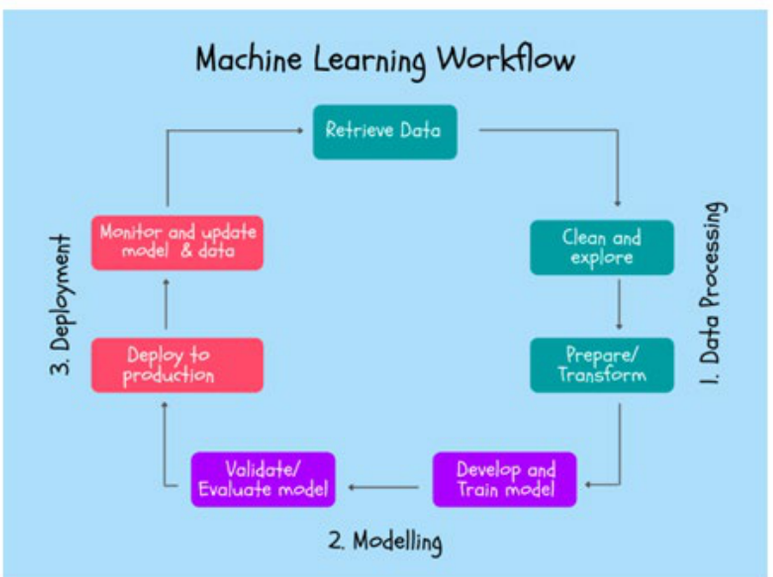


Fig. 2. Flujo de trabajo ML [16].

3.1. Tratamiento de los datos y entrenamiento del mejor modelo

El éxito de cualquier modelo de ML depende en gran medida de la calidad de los datos. Dado que no fue posible acceder a bases de datos reales de empresas constructoras ni realizar entrevistas exhaustivas con profesionales del sector en la provincia de Misiones, se recurrió al uso de conjuntos de datos sintéticos (Data set 1 y 2) y reales (Data set 3).

Data set 1: fue generado mediante un script en Python se generó 100.000 registros de 40 columnas, relacionados con proyectos de construcción, algunas de las características son: nombre

del constructor, código del constructor, código del proyecto, nombre del proyecto, año, ubicación, etapa, dimensión, tipo depropiedad, material nombre, material descripción, precio del material. Este conjunto incluye información simulada sobre la ubicación del proyecto, valoraciones del constructor, la seguridad, sociedad y amenidad relacionados a la ubicación de la propiedad. Se utilizó para pruebas iniciales y simulaciones [17].

Las características y descripción estadística del conjunto se pueden apreciar en la Fig. 3.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 40 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   builder_name          100000 non-null  object
 1   BuilderCode           100000 non-null  object
 2   projectCode           100000 non-null  object
 3   projectName           100000 non-null  object
 4   rateSqMeter           100000 non-null  int64
 5   year                  100000 non-null  int64
 6   location              100000 non-null  object
 7   stage                 100000 non-null  object
 8   dimension             100000 non-null  object
 9   propertyType          100000 non-null  object
 10  Flat_available        100000 non-null  int64
 11  totalOccupied_flat    100000 non-null  int64
 12  totalBooked           100000 non-null  int64
 13  onTime                100000 non-null  object
 14  expectedDeliveryDate  100000 non-null  object
 15  actualDeliveryDate    100000 non-null  object
 16  customerCode          100000 non-null  object
 17  projectCode.1         100000 non-null  object
 18  propertyType.1        100000 non-null  object
 19  customerName          100000 non-null  object
 20  customerEmail         100000 non-null  object
 21  customerPhone         100000 non-null  int64
 22  possessionDate        100000 non-null  object
 23  interiorQualityRating 100000 non-null  int64
 24  exteriorQualityRating 100000 non-null  int64
 25  societyRating         100000 non-null  int64
 26  securityRating        100000 non-null  int64
 27  amenities             100000 non-null  object
 28  BuilderRating         100000 non-null  float64
 29  materialCode          100000 non-null  object
 30  materialName          100000 non-null  object
 31  materialDescription   100000 non-null  object
 32  materialPrice         100000 non-null  float64
 33  date                  100000 non-null  object
 34  materialQty           100000 non-null  int64
 35  unit                  100000 non-null  object
 36  propertyCode          100000 non-null  object
 37  propertyAddress       100000 non-null  object
 38  propertyArea          100000 non-null  int64
 39  propertyPrice         100000 non-null  int64
dtypes: float64(2), int64(13), object(25)
memory usage: 30.5+ MB
```

```
data.describe()
```

	rateSqMeter	year	Flat_available	totalOccupied_flat	totalBooked	customerPhone	interiorQualityRating	exteriorQualityRating	societyRating
count	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	1.000000e+05	100000.00000	100000.000000	100000.000000
mean	9991.454380	2022.000710	50.035090	75.002700	25.014560	9.180009e+11	2.99421	3.001080	3.002750
std	2880.254115	1.415758	29.189854	14.682264	22.379293	1.157168e+09	1.41371	1.411481	1.413748
min	5000.000000	2020.000000	0.000000	50.000000	0.000000	9.160000e+11	1.00000	1.000000	1.000000
25%	7506.000000	2021.000000	25.000000	62.000000	7.000000	9.169952e+11	2.00000	2.000000	2.000000
50%	9984.000000	2022.000000	50.000000	75.000000	19.000000	9.180023e+11	3.00000	3.000000	3.000000
75%	12480.000000	2023.000000	75.000000	88.000000	39.000000	9.190023e+11	4.00000	4.000000	4.000000
max	15000.000000	2024.000000	100.000000	100.000000	100.000000	9.200000e+11	5.00000	5.000000	5.000000

Fig. 3. Características y descripción estadística del Data set 1.

Para identificar los datos relevantes, se procedió a listar las variables numéricas (medibles) y las variables categóricas (contextuales). Esto se aprecia en la Fig. 4.

```
numericas = data.select_dtypes(include=['int64', 'float64']).columns.tolist()
categoricas = data.select_dtypes(include=['object']).columns.tolist()

print("🔹 Variables numéricas (medibles):", len(numericas))
print(numericas)
print("\n🔸 Variables categóricas (contextuales):", len(categoricas))
print(categoricas)
```

```
🔹 Variables numéricas (medibles): 15
['rateSqMeter', 'year', 'Flat_available', 'totalOccupied_flat', 'totalBooked', 'customerPhone', 'interiorQualityRating', 'exteriorQualityRating', 'societyRating', 'securityRating', 'BuilderRating', 'materialPrice', 'materialQty', 'propertyArea', 'propertyPrice']

🔸 Variables categóricas (contextuales): 25
['builder_name', 'BuilderCode', 'projectCode', 'projectName', 'location', 'stage', 'dimension', 'propertyType', 'onTime', 'expectedDeliveryDate', 'actualDeliveryDate', 'customerCode', 'projectCode.1', 'propertyType.1', 'customerName', 'customerEmail', 'possessionDate', 'amenities', 'materialCode', 'materialName', 'materialDescription', 'date', 'unit', 'propertyCode', 'propertyAddress']
```

Fig. 4. Variables numéricas y variables categóricas del Data set 1.

Seguidamente, se procedió a obtener las respectivas distribuciones de las variables numéricas (Fig. 5) y la correlación entre las variables (Fig. 6).

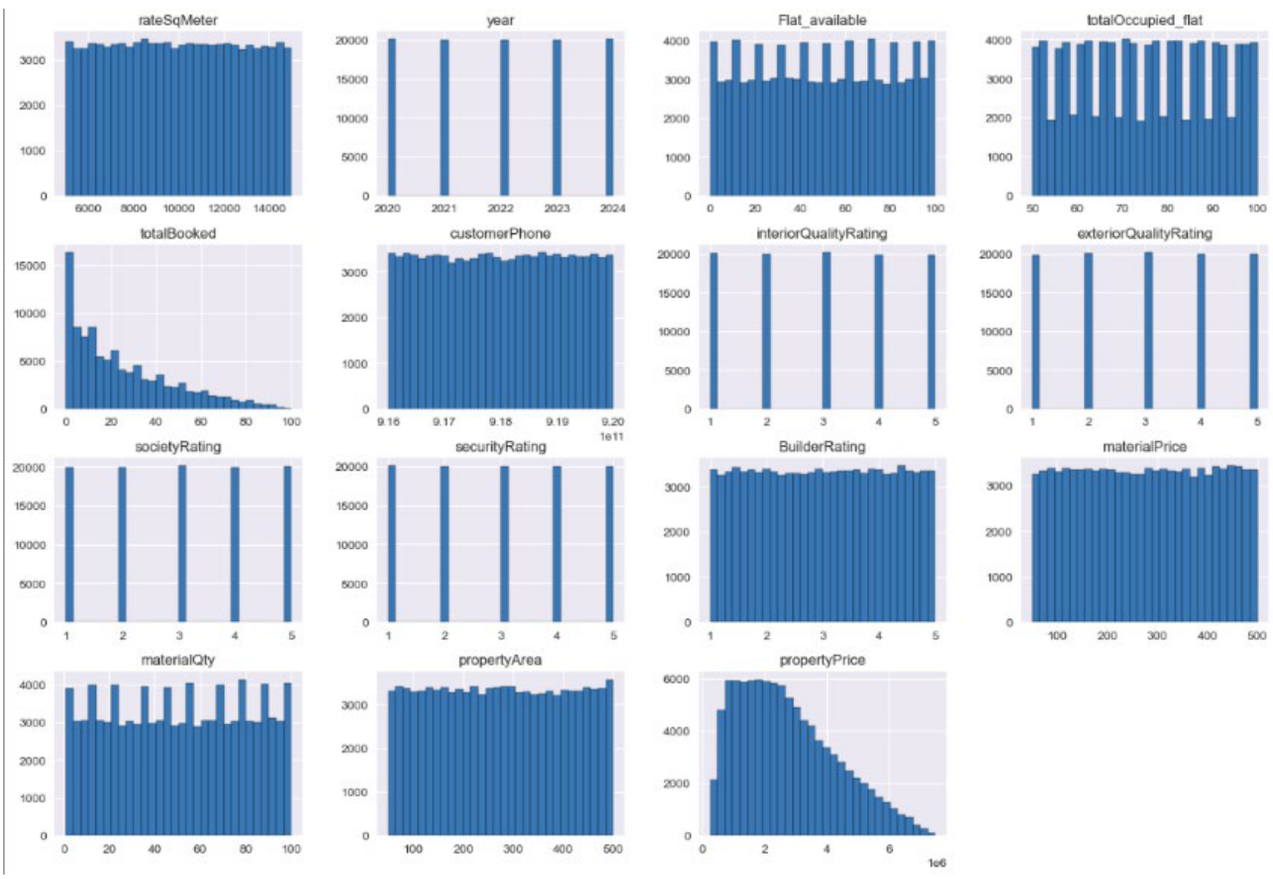


Fig. 5. Distribución estadística de las variables numéricas.

Se implementó la librería FLAML de AutoML para evaluar cuál es el mejor modelo de predicción según el set de datos; el algoritmo recomendado fue extra_tree con una exactitud del 99% (Fig. 7).

Se entrenó el modelo extra_tree con set de entrenamiento, ya que se dividió el data set en dos conjuntos. Set de entrenamiento con el 80% de los datos y set de prueba con el 20% los datos (Fig. 8).

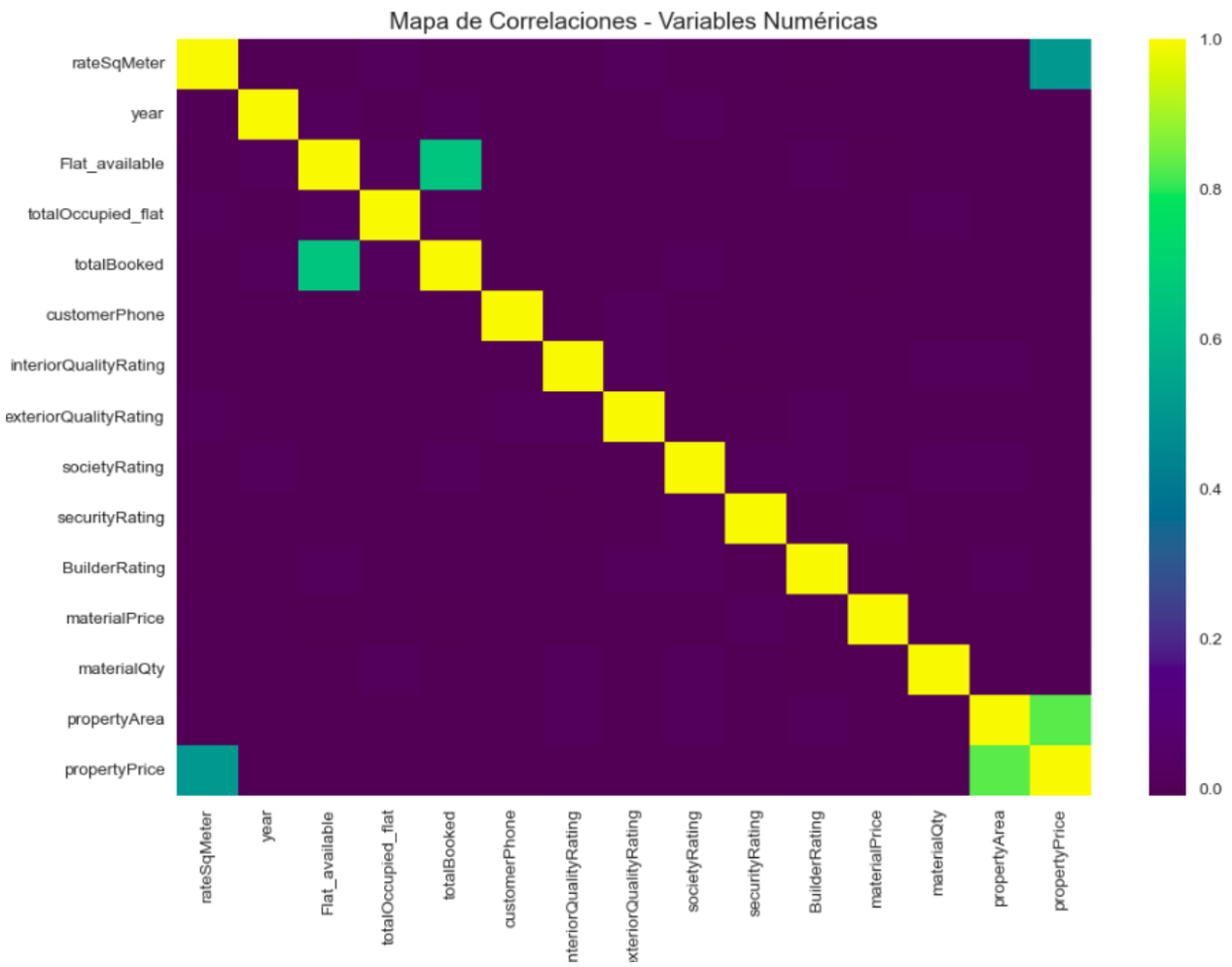


Fig. 6. Mapa de correlación entre variables.

```
[flaml.automl.logger: 10-18 22:43:06] {2724} INFO - retrain extra_tree for 20.7s
[flaml.automl.logger: 10-18 22:43:06] {2727} INFO - retrained model: ExtraTreesRegressor(max_leaf_nodes=7232, n_estimators=198, n_jobs=
1,
                    random_state=12032022)
[flaml.automl.logger: 10-18 22:43:06] {2009} INFO - fit succeeded
[flaml.automl.logger: 10-18 22:43:06] {2010} INFO - Time taken to find the best model: 83.58075213432312
🏆 Mejor modelo encontrado: extra_tree
R² en test: 0.9999939741040934
```

Fig. 7. Mejor modelo encontrado por AutoML (extra_tree).

```
📊 === EVALUACIÓN DEL MODELO ===
R² Train: 1.000000
R² Test:  0.927205
MAE Train: 0.00
MAE Test:  335,599.67
RMSE Train: 0.00
RMSE Test:  422,627.73
Error % Train: 0.00%
Error % Test:  19.70%
```

Fig. 8. Evaluación del modelo mediante MAE, R2 y RMSE.

Se aplicó validación cruzada (Fig. 9), para evaluar el modelo en múltiples divisiones de datos logrando una evaluación más robusta y evitando el *overfitting* . Los resultandos obtenidos fueron en promedio: 0.917701 esto explica el modelo con el 91.77% de varianza en promedio; con una

variabilidad: +/- 0.004138 muy baja siendo excelente para el modelo y la consistencia se refleja mediante el *folds* que están entre 0.914 - 0.920.

```
print("\n🔄 === VALIDACIÓN CRUZADA ===")
cv_scores = cross_val_score(extra_trees_model, X_train, y_train,
                            cv=5, scoring='r2', n_jobs=-1)

print(f"R² Validación Cruzada (5-fold): {cv_scores.mean():.6f} (+/- {cv_scores.std() * 2:.6f})")
print("Scores por fold:", [f"{score:.6f}" for score in cv_scores])
```

```
🔄 === VALIDACIÓN CRUZADA ===
R² Validación Cruzada (5-fold): 0.917701 (+/- 0.004138)
Scores por fold: ['0.917117', '0.920621', '0.919444', '0.916430', '0.914893']
```

Fig. 9. Validación cruzada.

Mediante el análisis residual (Fig. 10) se observó que en el gráfico de residuales vs. predicciones los puntos no siguen un patrón claro, sino que están distribuidos de manera aleatoria alrededor de la línea cero y existen algunos puntosmás dispersos en los valores centrales de las predicciones (posible problema). El gráfico predicciones vs valores reales muestra que el modelo tiene buena predicción, debido a que los puntos se agrupan de manera aceptable alrededor de la línea roja, asegurando una buena correlación del modelo. Y por último el gráfico distribución residual permite interpretar que la distribución es normal, la mediana está cerca de cero, la forma es simétrica con una leve asimetría hacia la izquierda.

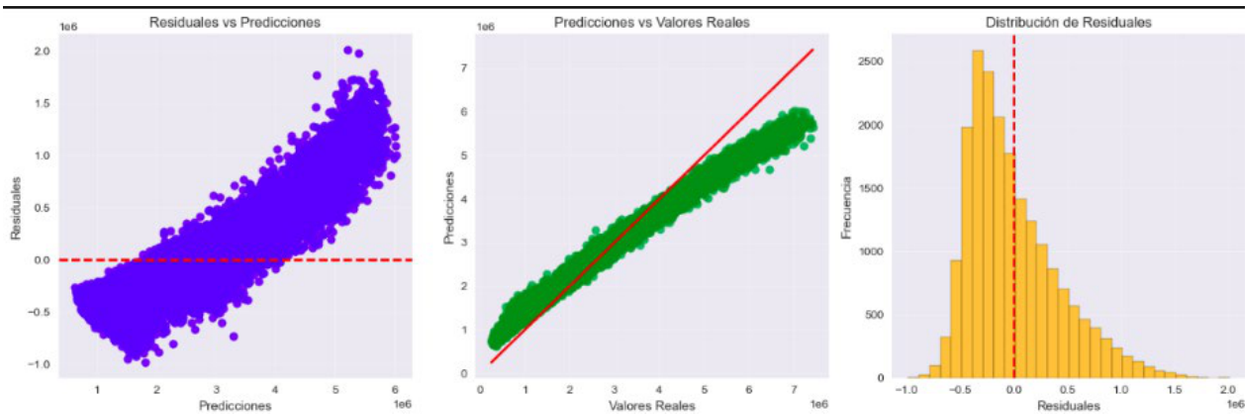


Fig. 10. Resultado análisis residual.

Los resultados obtenidos al evaluar el modelo con el set de test permiten describir que el R^2 Score fue de 0.9272, esto significa un excelente rendimiento, dado que el modelo logra explicar el 92.72% de la variabilidad en los datos. Los errores absoluto promedio (error cuadrático medio RMSE = 422,627.73 y error absoluto MAE: 335,599.67) muestran que las predicciones se desvían en promedio $335,600 del valor real, en este punto se deberá considerar el rango típico de los valores para así evaluar si esto es bueno o malo.

El gráfico de la figura 11 , muestra la Distribución Asimétrica Positiva Mediana (11.14%) < Promedio (19.70%) indicando la existencia de *outliers* que inflan el promedio. Además,se observa que el 25% de las predicciones tienen error ≤ 5.35% (muy bueno) y el 75% de las predicciones tienen error ≤ 21.66% (3 de cada 4 casos). La diferencia que se observa entre el Promedio= 19.70% y la Mediana = 11.14% significa que hay predicciones con errores muy altos que elevan el

promedio; por lotanto, detectando y eliminando los *outliers*, el error típico sería alrededor del 11%.

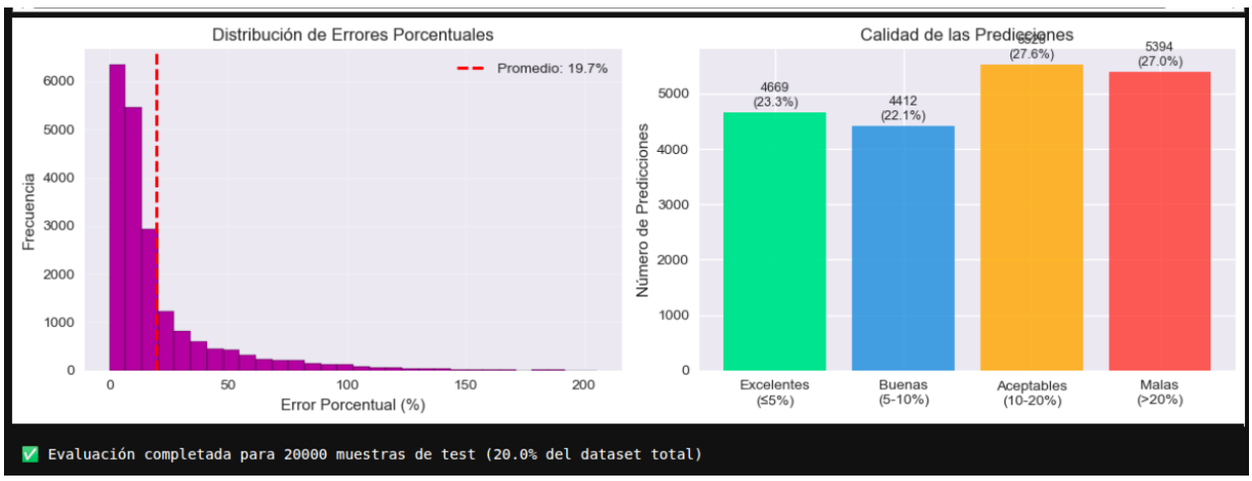


Fig. 11. Evaluación del modelo con el set test (20000 ejemplos).

Por lo tanto, para el set de datos 1 el modelo extra_tree podrá estimar el costo de un proyecto con una capacidad explicativa excelente de $R^2 = 0.927$; el 50% de las predicciones con error ≤ 10% y el 75% de casos con error ≤ 21.66%. Ofreciendo un buen rendimiento general para la mayoría de casos, siendo confiable para 73% de los casos aproximadamente y es problemático para el 27% de los casos restantes.

A continuación, se buscó el mejor modelo para estimar el tiempo utilizando el mismos Data set 1. Los pasos que se siguieron fueron: preprocesamiento de los datos para descartar características no relevantes para el modelo, selección del modelo, entrenamiento y prueba del modelo. Posterior a la limpieza de los datos el data set que se obtuvo se muestra en la Fig. 12.

En la Fig. 13a se muestra el modelo con mayor precisión para estimar el tiempo de finalización de un proyecto de construcción, nuevamente utilizamos la librería FLAML de AutoML. El modelo seleccionado fue*xgb_limitdepth* que es el modelo XGBoost con profundidad limitada, dando como salida R^2 en test = - 0.00028, lo cual indica un grave error y es que el modelo no está aprendiendo. Como solución al problema detectado, se aplicó ingeniería de características (Fig. 13b), para definir un target más robusto, filtrar valores extremos, seleccionar características potencialmente relevantes y convertir algunas categorías, antes de aplicar nuevamente AutoML.

Los resultados preliminares (Fig. 14) evidencian un desempeño insuficiente, por lo que se propone realizar un análisis exploratorio más exhaustivo de los datos con el fin de identificar posibles *outliers* y mejorar la calidad del modelo.

```
Preprocesando datos...
Valores nulos por columna:
builder_name             0
BuilderCode              0
projectCode              0
projectName              0
rateSqMeter              0
year                     0
location                 0
stage                    0
dimension                0
propertyType             0
Flat_available           0
totalOccupied_flat       0
totalBooked              0
onTime                   0
expectedDeliveryDate     0
actualDeliveryDate       0
customerCode             0
projectCode.1            0
propertyType.1           0
customerName             0
customerEmail            0
customerPhone            0
possessionDate           0
interiorQualityRating    0
exteriorQualityRating    0
societyRating            0
securityRating           0
amenities                0
BuilderRating            0
materialCode             0
materialName             0
materialDescription      0
materialPrice            0
date                     0
materialQty              0
unit                     0
propertyCode             0
propertyAddress          0
propertyArea             0
propertyPrice            0
dtype: int64
Tamaño del dataset después de limpieza: (100000, 41)
```

Fig. 12a. Preprocesamiento de datos.

```python
# Paso 5: Selección de características
# Columnas categóricas para codificar
categorical_features = [
    'builder_name', 'location', 'stage', 'dimension', 'propertyType',
    'onTime', 'amenities', 'unit'
]

# Columnas numéricas
numerical_features = [
    'rateSqMeter', 'year', 'Flat_available', 'totalOccupied_flat',
    'totalBooked', 'interiorQualityRating', 'exteriorQualityRating',
    'societyRating', 'securityRating', 'BuilderRating', 'materialPrice',
    'materialQty', 'propertyArea', 'propertyPrice'
]

# Codificar variables categóricas
label_encoders = {}
for col in categorical_features:
    if col in df.columns:
        le = LabelEncoder()
        df[col + '_encoded'] = le.fit_transform(df[col].astype(str))
        label_encoders[col] = le

# Crear conjunto final de características
feature_columns = [col + '_encoded' for col in categorical_features if col in df.columns] + numerical_features

# Eliminar columnas con valores nulos en las características
df_clean = df[feature_columns + ['days_until_delivery']].dropna()

print(f"Tamaño final del dataset: {df_clean.shape}")
```

```
Tamaño final del dataset: (100000, 23)
```

Fig. 12b. Selección de características.

```
            monotone_constraints=None, multi_strategy=None, n_estimators=8,
            n_jobs=-1, num_parallel_tree=None, ...)
[flaml.automl.logger: 10-19 12:11:13] {2009} INFO - fit succeeded
[flaml.automl.logger: 10-19 12:11:13] {2010} INFO - Time taken to find the best model: 50.822264432907104
🏆 Mejor modelo encontrado: xgb_limitdepth
R² en test: -0.0002887007311913603
```

Fig. 13a. Selección de modelo para estimar el tiempo de finalización.

```python
automl = AutoML()

automl_settings = {
    "time_budget": 300,  # 5 minutos
    "metric": 'r2',
    "task": 'regression',
    "estimator_list": ['lgbm', 'xgboost', 'rf'],  # Forzar algoritmos potentes
    "seed": 42,
    "verbose": 2
}

print("🔍 Entrenando modelo mejorado...")
automl.fit(X_train=X_train, y_train=y_train, **automl_settings)

# Resultados
print("\n" + "="*50)
print("🏆 MEJOR MODELO:", automl.best_estimator)
y_pred = automl.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("📊 R² en test:", r2)

# Diagnóstico
if r2 > 0.3:
    print("✅ ¡MEJORA SIGNIFICATIVA!")
elif r2 > 0:
    print("⚠️  Modelo mejorado pero necesita trabajo")
else:
    print("🔴 El problema persiste - necesita análisis más profundo")
```

```
🎯 19 características seleccionadas
📊 Tamaño del dataset: 90423 proyectos
🔍 Entrenando modelo mejorado...

==================================================
🏆 MEJOR MODELO: lgbm
📊 R² en test: -0.0005547782032857551
🔴 El problema persiste - necesita análisis más profundo
```

Fig. 13b. Selección del modelo posterior a la Ingeniería de características.

```
----------------------------------------
Estadísticas de 'days_until_completion':
• Media: 197.6 días
• Mediana: 197.0 días
• Mínimo: 30 días
• Máximo: 365 días
• Desviación estándar: 97.1 días
• Coeficiente de variación: 49.1%

Text(0, 0.5, 'Días')
```

Fig. 14a. Análisis de variable objetivo.

```
----------------------------------------
• Límite inferior: -139.0 días
• Límite superior: 533.0 días
• Número de outliers: 0 (0.0%)
]: Text(0, 0.5, 'Días')
```

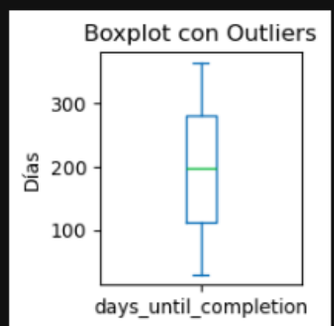


Fig. 14b. Valores extremos y *outliers*.

```
----------------------------------------
Top 10 correlaciones con días hasta finalización:
                correlation  abs_correlation
securityRating     0.007750         0.007750
societyRating      0.007384         0.007384
BuilderRating     -0.004357         0.004357
materialQty       -0.003797         0.003797
rateSqMeter        0.003328         0.003328
propertyPrice      0.003119         0.003119
Flat_available    -0.002153         0.002153
propertyArea       0.001825         0.001825
materialPrice      0.001274         0.001274
totalBooked       -0.001138         0.001138
Text(0.5, 0, 'Coeficiente de Correlación')
```

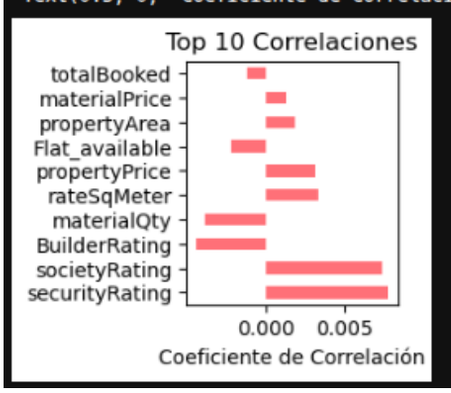


Fig. 14c. Análisis de correlaciones.

```
----------------------------------------
Top 10 características por información mutua:
propertyArea              0.003917
rateSqMeter               0.002926
BuilderRating             0.002093
materialQty               0.001967
Flat_available            0.001537
materialPrice             0.001296
exteriorQualityRating     0.000758
year                      0.000000
totalOccupied_flat        0.000000
totalBooked               0.000000
dtype: float64
```

Fig. 14d. Análisis de información que comparten 2 variables.

Este análisis evidencia que el **Data set 1** contiene características que no presentan relación alguna con el tiempo de finalización del proyecto. Tanto las correlaciones como la información mutua resultan prácticamente nulas, lo que indica que las variables disponibles no aportan valor predictivo. Si bien la variable objetivo *days_until_completion* presenta valores entre 30 y 365 días, dichos tiempos no dependen de las características del proyecto.

Además, al tratarse de datos sintéticos o generados artificialmente, no reflejan relaciones reales propias del proceso constructivo y carecen de información crucial (como condiciones climáticas, permisos, mano de obra o capacidad del equipo). Ejemplos evidentes de esto son que un proyecto de 50 m² y otro de 500 m² registran exactamente el mismo tiempo de ejecución, o que un contratista con *rating* 1 finaliza en el mismo plazo que uno con *rating* 5.

Por lo tanto, se infiere que los tiempos de entrega se mantienen fijos o estandarizados, independientemente de las características consideradas. Las correlaciones observadas resultan muy débiles y las variables disponibles no aportan información predictiva relevante. En consecuencia, se decide trabajar con otro conjunto de datos, dado que este se considera de calidad insuficiente para abordar adecuadamente el problema planteado.

Data set 2: este set de datos se encuentra en [18], está compuesto por 300 ejemplos y 15 características. En la Fig. 15a y 15b se exhibe la descripción estadística de las características.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Project ID      300 non-null    object
 1   Project Name    300 non-null    object
 2   Project Type    300 non-null    object
 3   Location        300 non-null    object
 4   Start Date      300 non-null    object
 5   End Date        300 non-null    object
 6   Project Status  300 non-null    object
 7   Priority        300 non-null    object
 8   Task ID         300 non-null    object
 9   Task Name       300 non-null    object
 10  Task Status     300 non-null    object
 11  Assigned To     300 non-null    object
 12  Hours Spent     300 non-null    int64
 13  Budget          300 non-null    int64
 14  Actual Cost     300 non-null    int64
 15  Progress        300 non-null    float64
dtypes: float64(1), int64(3), object(12)
memory usage: 37.6+ KB
```

Fig. 15a– Descripción de los ejemplos y características

df.describe()

	Hours Spent	Budget	Actual Cost	Progress
count	300.000000	300.000000	300.000000	300.000000
mean	19.210000	5624.266667	1728.070000	0.636067
std	11.505382	2706.584211	3012.743452	0.336683
min	0.000000	1026.000000	0.000000	0.000000
25%	10.000000	3312.000000	0.000000	0.352500
50%	18.500000	5751.500000	0.000000	0.680000
75%	29.000000	8287.750000	2667.000000	1.000000
max	40.000000	9998.000000	12398.000000	1.000000

Fig. 15b– Descripción estadística.

Para seleccionar el modelo más adecuado para predecir el tiempo de finalización de un proyecto de construcción, se identificaron las características categóricas más relevantes y se transformaron a variables numéricas mediante técnicas de codificación apropiadas.

Con la librería ProfileReport se realizó un análisis exploratorio (Fig. 16a y 16b), el cual ofrece como salida un informe interactivo (en HTML o dentro de Jupyter Notebook) que resume toda la información estadística y estructural del set de datos (como valores faltantes y correlación entre variables).

Project Name

Text

Distinct	100
Distinct (%)	33.3%
Missing	0
Missing (%)	0.0%
Memory size	2.5 KiB

Fig. 16a. Información de las características (ProfileReport).

Overview | Alerts 5 | Reproduction

Dataset statistics

Number of variables	16
Number of observations	300
Missing cells	203
Missing cells (%)	4.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	37.6 KiB
Average record size in memory	128.4 B

Variable types

Text	3
Categorical	7
DateTime	2
Numeric	4

Fig. 16b. Información estadística del data set (ProfileReport).

Seguidamente se realizó ingeniería de características para determinar las características más relevantes para el escenario planteado (Fig. 17).

```
# 5. Importancia de características
importancias = model.feature_importances_
feature_importance = pd.DataFrame({'Característica': features, 'Importancia': importancias})
feature_importance = feature_importance.sort_values('Importancia', ascending=False)

print("🔍 Características más relevantes para predecir días restantes:")
print(feature_importance)
```

```
🔍 Características más relevantes para predecir días restantes:
        Característica  Importancia
0  duracion_planificada     0.911247
1              Progress     0.088753

evaluacion del modelo
```

Fig. 17. Ingeniería de características.

Se seleccionó el modelo Random Forest Regressor (siguiendo lo sugerido en la bibliografía). Para el entrenamiento se utilizó 80% de los ejemplos del data set, dejando el 20% de los ejemplos para las pruebas del modelo (Fig. 18). Los resultados obtenidos del entrenamiento del modelo se muestran en la Figura 19a y 19b.

```
# 3. Dividir datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 4. Entrenar modelo
model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Fig. 18. Entrenamiento del modelo

Los resultados obtenidos del entrenamiento del modelo se muestran en la Figura 19a y 19b.

```python
# 5. Predecir y evaluar
y_pred = model.predict(X_test)

# Métricas de evaluación
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("📊 Métricas de Evaluación:")
print(f"- MAE (Error Absoluto Medio): {mae:.2f} días")
print(f"- MSE (Error Cuadrático Medio): {mse:.2f}")
print(f"- RMSE (Raíz del Error Cuadrático Medio): {rmse:.2f} días")
print(f"- R² (Coeficiente de Determinación): {r2:.2f}")
```

```
📊 Métricas de Evaluación:
- MAE (Error Absoluto Medio): 60.32 días
- MSE (Error Cuadrático Medio): 5286.07
- RMSE (Raíz del Error Cuadrático Medio): 72.71 días
- R² (Coeficiente de Determinación): 0.77
```

Fig. 19a. Evaluación y predicción del modelo.

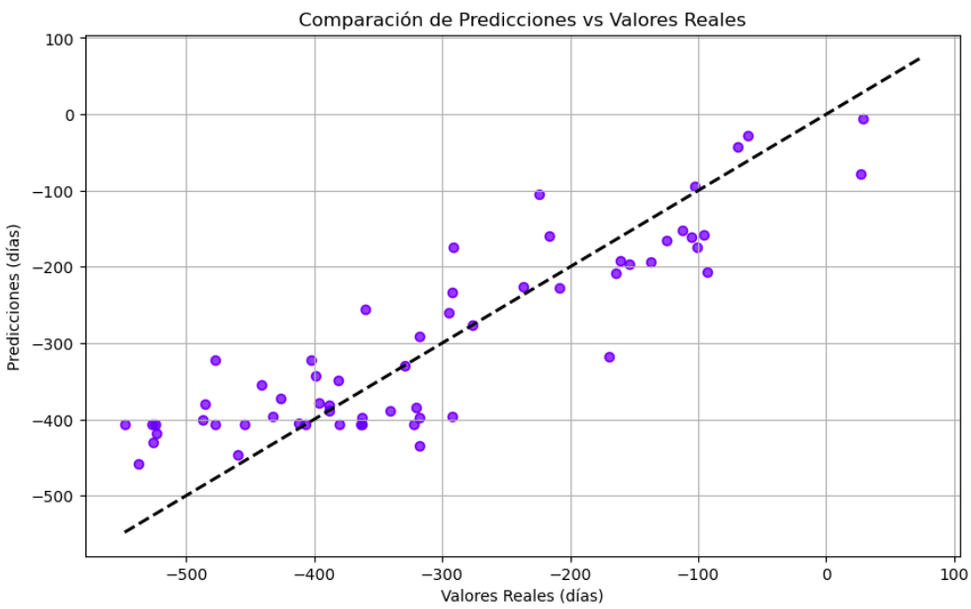


Fig. 19b. Correlación del modelo con datos de entrenamiento.

Asimismo, este data set se utilizó para entrenar y evaluar el modelo XGBoost, los resultados no mejoraron, en la Figura 20 se muestran las métricas obtenidas.

```python
# 6. Predecir y evaluar
y_pred = best_xgb.predict(X_test_scaled)

# Métricas de evaluación
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\n📊 Métricas de Evaluación (XGBoost):")
print(f"- MAE (Error Absoluto Medio): {mae:.2f} días")
print(f"- MSE (Error Cuadrático Medio): {mse:.2f}")
print(f"- RMSE (Raíz del Error Cuadrático Medio): {rmse:.2f} días")
print(f"- R² (Coeficiente de Determinación): {r2:.2f}")
```

```
📊 Métricas de Evaluación (XGBoost):
- MAE (Error Absoluto Medio): 60.10 días
- MSE (Error Cuadrático Medio): 5289.55
- RMSE (Raíz del Error Cuadrático Medio): 72.73 días
- R² (Coeficiente de Determinación): 0.77
```

Fig. 20. Entrenamiento y prueba del modelo XGBoost.

En cuanto a la predicción del costo se definió como target a la característica *Actual_Cost* y se seleccionó el modelo XGBRegressor. Las métricas obtenidas se pueden apreciar en la Fig. 21.

```
# ====================
# 5. Evaluar
# ====================
y_pred = model.predict(X_test)
print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
print(f"RMSE: {mean_squared_error(y_test, y_pred):.2f}")
print(f"R²: {r2_score(y_test, y_pred):.2f}")
```

```
MAE: 314.01
RMSE: 965121.70
R²: 0.89
```

Fig. 21. Métricas del modelo XGBRegressor.

Con el objetivo de mejorar el error cuadrático se decidió determinar las características más relevantes, para luego entrenar y probar el mismo modelo (Fig. 22a y 22b).

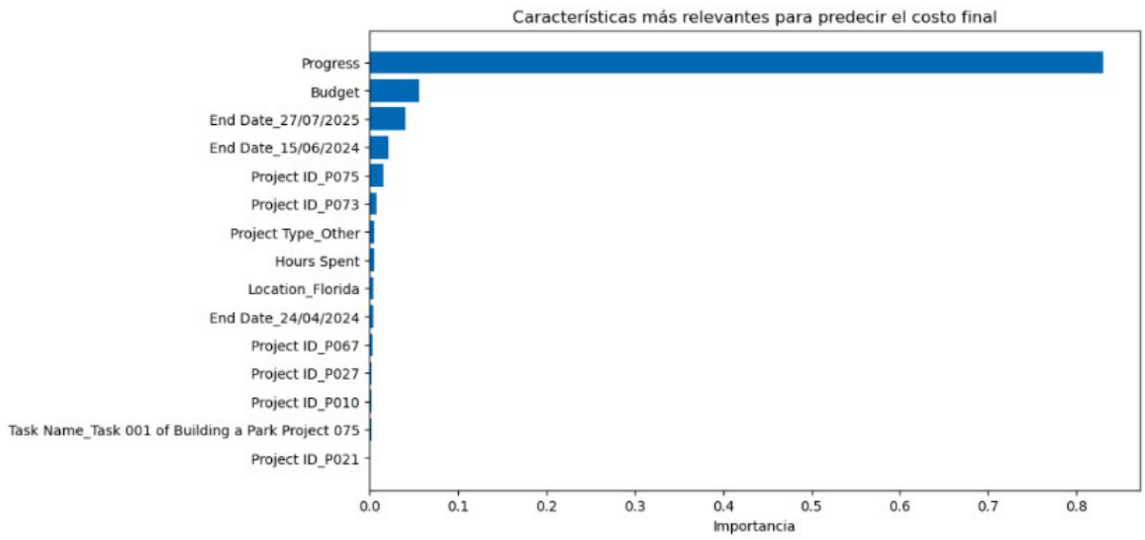


Fig. 22a. Características relevantes para estimar el costo.

Seguidamente, se probó y evaluó el modelo utilizando únicamente las características relevantes, logrando una mejora del 1 % respecto de las métricas obtenidas previamente (Fig. 23).

```
# ====================
# 7. Imprimir ranking
# ====================
feature_ranking = pd.DataFrame({
    "Feature": feature_names[indices],
    "Importance": importances[indices]
})
print(feature_ranking.head(15))
```

```
                                        Feature  Importance
0                                      Progress    0.830811
1                                        Budget    0.056030
2                           End Date_27/07/2025    0.040093
3                           End Date_15/06/2024    0.020782
4                               Project ID_P075    0.014862
5                               Project ID_P073    0.007075
6                            Project Type_Other    0.005103
7                                   Hours Spent    0.004854
8                              Location_Florida    0.004306
9                           End Date_24/04/2024    0.003837
10                              Project ID_P067    0.003218
11                              Project ID_P027    0.001864
12                              Project ID_P010    0.001725
13  Task Name_Task 001 of Building a Park Project 075    0.001702
14                              Project ID_P021    0.001093
```

Fig. 22b. Ranking de características.

```
# ====================
# 3. Entrenamiento inicial para obtener importancia
# ====================
model_init = xgb.XGBRegressor(
    n_estimators=200,
    learning_rate=0.05,
    max_depth=6,
    random_state=42
)
model_init.fit(X_scaled, y)

# Obtener importancia
importances = model_init.feature_importances_
feature_names = X_encoded.columns
importance_df = pd.DataFrame({
    "Feature": feature_names,
    "Importance": importances
}).sort_values(by="Importance", ascending=False)

# Seleccionar características que sumen hasta el 95% de la importancia
importance_df["Cumulative"] = importance_df["Importance"].cumsum()
selected_features = importance_df[importance_df["Cumulative"] <= 0.95]["Feature"].tolist()

print(f"Se seleccionaron {len(selected_features)} características más relevantes.")
```

```
Se seleccionaron 4 características más relevantes.
```

```
# ====================
# 4. Filtrar dataset con las mejores features
# ====================
X_selected = X_encoded[selected_features]
X_train, X_test, y_train, y_test = train_test_split(
    X_selected, y, test_size=0.2, random_state=42
)
```

Fig. 23. Filtrado de las características relevantes para estimar el costo.

Los resultados del entrenamiento del modelo XGBRegressor utilizando las cuatro características más relevantes se muestran en la Figura 24.

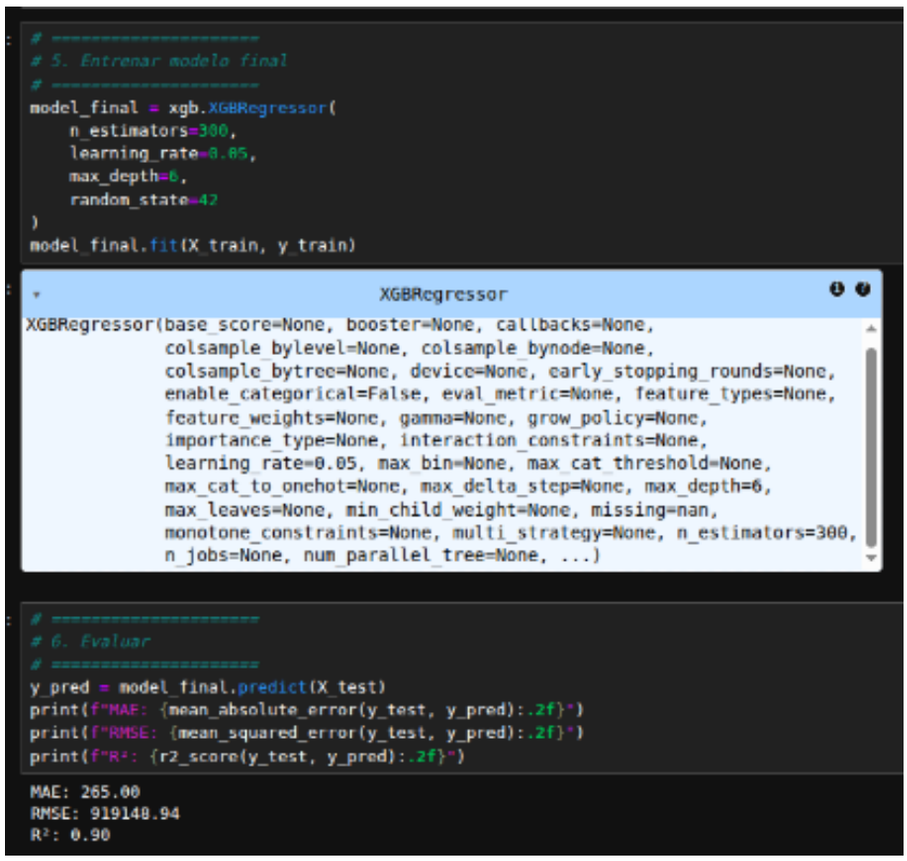


Fig. 24. Métricas de XGBRegressor.

Data set 3: compuesto por 378 ejemplos con 109 características relacionados a proyectos de construcción [19].

Para evaluar la viabilidad de utilizar este conjunto de datos en el proyecto desarrollado, se recurrió a una consulta mediante NotebookLM, a fin de obtener un análisis inicial sobre su pertinencia y calidad [20]. El objetivo fue evaluar la relevancia del data set para estimar tiempo y costo en la finalización de proyectos de construcción. La salida proporcionada fue un documento con una gran cantidad de datos relacionados con proyectos de construcción residencial, organizados según el calendario persa, con columnas para el año/trimestre de inicio del proyecto y el año/trimestre de finalización del proyecto. Una sección significativa de la información se centra en variables físicas y financieras del proyecto (V-1 a V-10), incluyendo medidas de área como metros cuadrados (m^2) y valores monetarios en la moneda iraní (Rial). Además, el texto enumera extensas variables e índices económicos (V-11 a V-29), como el índice de precios al por mayor (WPI), la liquidez acumulada, y varios índices de precios al consumidor (IPC), presentados con hasta cinco retrasos de tiempo para el análisis. Finalmente, el pie de página aclara que algunas variables se han calibrado entre 0 y 1 y proporciona definiciones para los identificadores de las variables.

La preparación de los datos para el modelado consistió en cargar el conjunto de datos omitiendo las filas de encabezado incorrectas, para luego identificar la estructura real de la información. A partir de ello, se construyó un nuevo *Data Frame* con la organización adecuada, se determinaron las columnas clave y se analizaron patrones en su nomenclatura. Finalmente, se asignaron nombres más claros y consistentes, atendiendo a la estructura del conjunto de datos: columnas 0–3

correspondientes a fechas del proyecto; columnas 4–11 asociadas a variables físicas y financieras (V-1 a V-8); y las dos últimas columnas correspondientes a las variables objetivo (V-9 y V-10).

La variable objetivo (target) seleccionada para la estimación del tiempo de ejecución del proyecto corresponde a la característica V-7 del conjunto de datos, denominada **"Duration of Construction"**, la cual presenta un tipo de dato numérico. Por otro lado, para la predicción del gasto total del proyecto, se definió como target la característica V-10, asociada a **"Actual Construction Costs (output)"**, es decir, el costo real incurrido durante la construcción.

La selección de ambas variables se sustenta en su relación directa con los objetivos planteados en esta etapa del estudio (Figs. 25a, 25b, 25c y 25d).

```
Datos cargados correctamente. Dimensiones del DataFrame de datos: (377, 109)

Primeras 5 filas del DataFrame de datos:
   START YEAR  START QUARTER  COMPLETION YEAR  COMPLETION QUARTER  V-1     V-2  \
0          81            1.0             85.0                 1.0  1.0  3150.0
1          84            1.0             89.0                 4.0  1.0  7600.0
2          78            1.0             81.0                 4.0  1.0  4800.0
3          72            2.0             73.0                 2.0  1.0   685.0
4          87            1.0             90.0                 2.0  1.0  3000.0

      V-3     V-4    V-5      V-6   V-7     V-8     V-11   V-12    V-13  V-14  \
0   920.0   598.5  190.0  1010.84  16.0  1200.0  6713.00   56.2   61.52  6.11
1  1140.0  3040.0  400.0   963.81  23.0  2900.0  3152.00  106.0  103.03  3.15
2   840.0   480.0  100.0   689.84  15.0   630.0  1627.00   41.0   41.25  1.74
3   202.0    13.7   20.0   459.54   4.0   140.0  2580.93   12.1   10.03  1.24
4   800.0  1230.0  410.0   631.91  13.0  5000.0  6790.00  203.8  162.84  6.46

        V-15     V-16   V-17    V-18     V-19  V-20    V-21    V-22     V-23  \
```

Fig. 25a. Data Frame.

```
--- Nombres de TODAS las columnas en df_data ---
['START YEAR', 'START QUARTER', 'COMPLETION YEAR', 'COMPLETION QUARTER', 'V-1', 'V-2', 'V-3', 'V-4', 'V-5', 'V-6', 'V-7', 'V-8', 'V-11', 'V-
12', 'V-13', 'V-14', 'V-15', 'V-16', 'V-17', 'V-18', 'V-19', 'V-20', 'V-21', 'V-22', 'V-23', 'V-24', 'V-25', 'V-26', 'V-27', 'V-28', 'V-29',
'V-11.1', 'V-12.1', 'V-13.1', 'V-14.1', 'V-15.1', 'V-16.1', 'V-17.1', 'V-18.1', 'V-19.1', 'V-20.1', 'V-21.1', 'V-22.1', 'V-23.1', 'V-24.1',
'V-25.1', 'V-26.1', 'V-27.1', 'V-28.1', 'V-29.1', 'V-11.2', 'V-12.2', 'V-13.2', 'V-14.2', 'V-15.2', 'V-16.2', 'V-17.2', 'V-18.2', 'V-19.2',
'V-20.2', 'V-21.2', 'V-22.2', 'V-23.2', 'V-24.2', 'V-25.2', 'V-26.2', 'V-27.2', 'V-28.2', 'V-29.2', 'V-11.3', 'V-12.3', 'V-13.3', 'V-14.3',
'V-15.3', 'V-16.3', 'V-17.3', 'V-18.3', 'V-19.3', 'V-20.3', 'V-21.3', 'V-22.3', 'V-23.3', 'V-24.3', 'V-25.3', 'V-26.3', 'V-27.3', 'V-28.3',
'V-29.3', 'V-11.4', 'V-12.4', 'V-13.4', 'V-14.4', 'V-15.4', 'V-16.4', 'V-17.4', 'V-18.4', 'V-19.4', 'V-20.4', 'V-21.4', 'V-22.4', 'V-23.4',
'V-24.4', 'V-25.4', 'V-26.4', 'V-27.4', 'V-28.4', 'V-29.4', 'V-9', 'V-10']
----------------------------------------
```

Fig. 25b. Encabezado de las características.

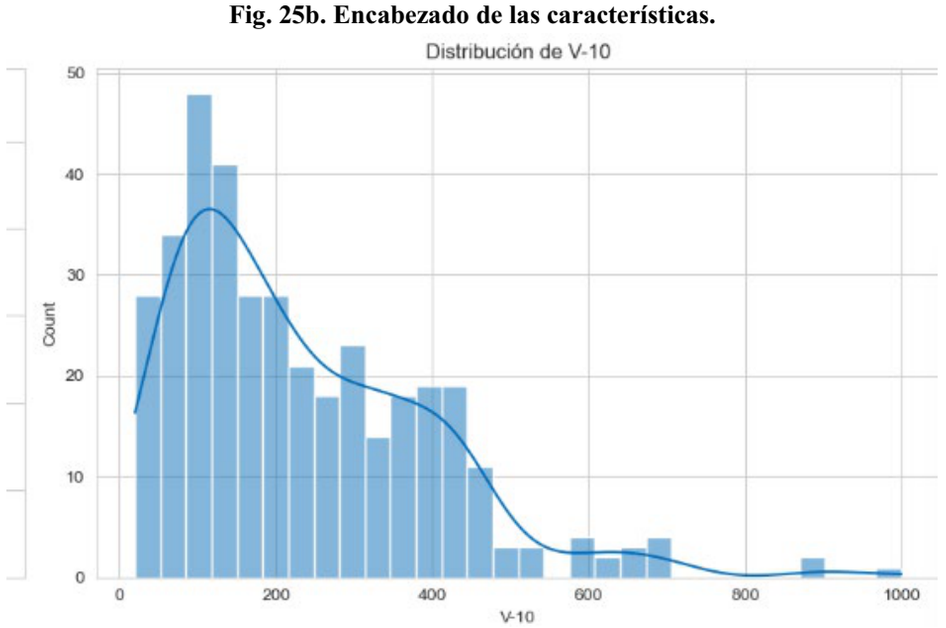


Fig. 25c. Distribución estadística target V-10.

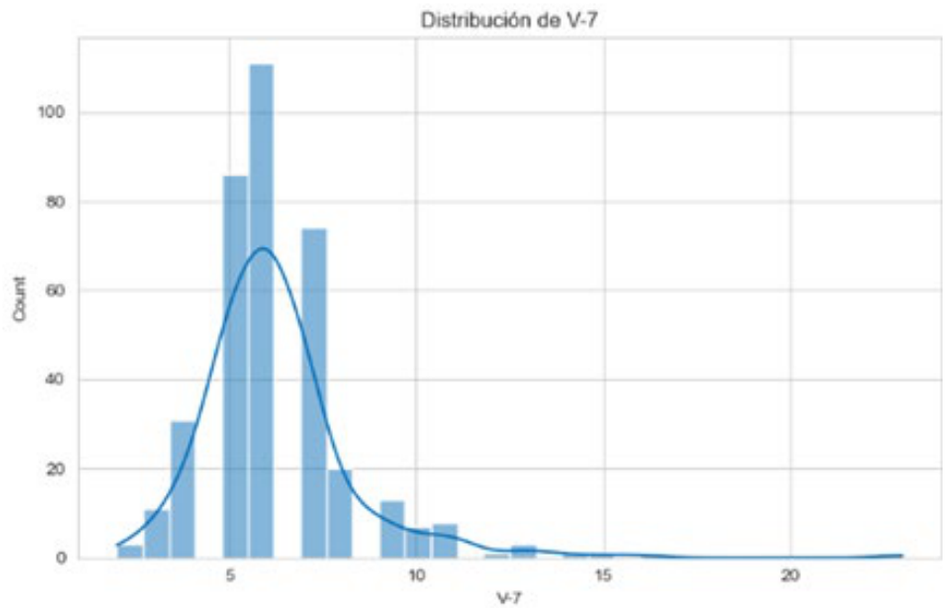


Fig. 25d. Distribución estadística target V-7.

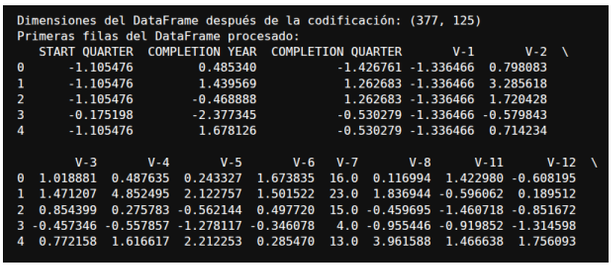
Posteriormente, se llevó a cabo el preprocesamiento del conjunto de datos, el cual incluyó la identificación y clasificación de las características en categóricas y numéricas, la detección de valores nulos o faltantes, y el análisis de correlaciones entre variables para evaluar relaciones significativas. Las variables categóricas fueron transformadas en características numéricas mediante la técnica *One-Hot Encoding*, empleando la librería Pandas, que genera una nueva columna para cada categoría presente en la variable original. Asimismo, se realizó la estandarización de las características numéricas, procedimiento fundamental para garantizar un desempeño adecuado en numerosos algoritmos de ML al homogenizar las escalas de los datos. Este proceso de preparación se resume en las Figs. 26a, 26b, 26c y 26d.

```
Correlación FUERTE (≥ 0.5) con las variables target:

--- Correlación con V-7 ---
No hay variables con correlación absoluta ≥ 0.5

--- Correlación con V-10 ---
V-5       0.963322
V-9       0.796041
V-8       0.789944
V-13.3    0.780323
V-25.3    0.780152
             ...
V-24.2    0.606195
V-24      0.604706
V-4       0.602107
V-19.3    0.566040
V-20     -0.521631
Name: V-10, Length: 76, dtype: float64
```

Fig. 26a. Correlaciones fuertes entre características.

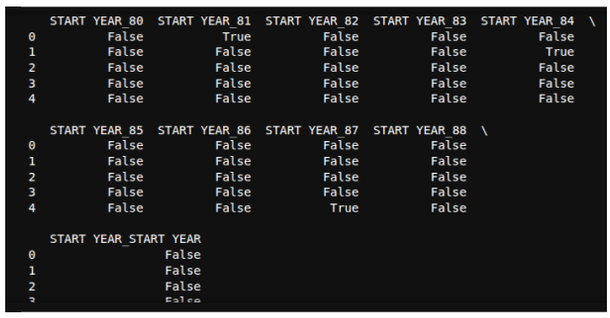
```
--- INICIANDO LIMPIEZA ADICIONAL DE VARIABLES TARGET ---
Filas originales: 377
Filas eliminadas (donde TARGET_TIME o TARGET_COST era NaN): 5
Filas finales en df_data: 372
NaN restantes en V-7: 0
NaN restantes en V-10: 0
```

Fig. 26b. Eliminación de valores nulos.



```
Dimensiones del DataFrame después de la codificación: (377, 125)
Primeras filas del DataFrame procesado:
   START QUARTER  COMPLETION YEAR  COMPLETION QUARTER      V-1       V-2  \
0      -1.105476         0.485340           -1.426761 -1.336466  0.798083
1      -1.105476         1.439569            1.262683 -1.336466  3.285618
2      -1.105476        -0.468888            1.262683 -1.336466  1.720428
3      -0.175198        -2.377345           -0.530279 -1.336466 -0.579843
4      -1.105476         1.678126           -0.530279 -1.336466  0.714234

       V-3       V-4       V-5       V-6   V-7       V-8      V-11      V-12  \
0  1.018881  0.487635  0.243327  1.673835  16.0  0.116994  1.422980 -0.608195
1  1.471207  4.852495  2.122757  1.501522  23.0  1.836944 -0.596062  0.189512
2  0.854399  0.275783 -0.562144  0.497720  15.0 -0.459695 -1.460718 -0.851672
3 -0.457346 -0.557857 -1.278117 -0.346078   4.0 -0.955446 -0.919852 -1.314598
4  0.772158  1.616617  2.212253  0.285470  13.0  3.961588  1.466638  1.756093
```

Fig. 26c. Estandarización de características numéricas.

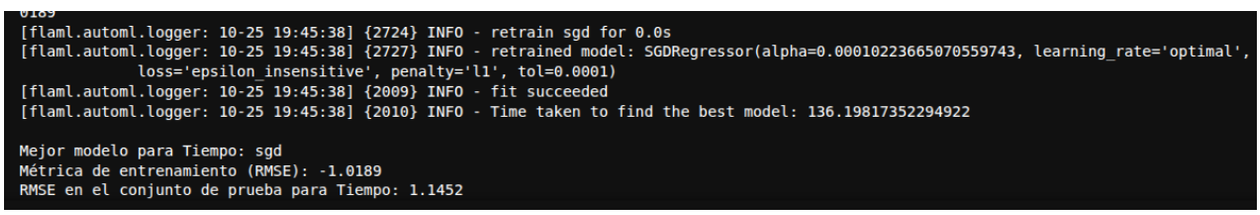


```
   START YEAR_80  START YEAR_81  START YEAR_82  START YEAR_83  START YEAR_84  \
0          False           True          False          False          False
1          False          False          False          False           True
2          False          False          False          False          False
3          False          False          False          False          False
4          False          False          False          False          False

   START YEAR_85  START YEAR_86  START YEAR_87  START YEAR_88  \
0          False          False          False          False
1          False          False          False          False
2          False          False          False          False
3          False          False          False          False
4          False          False           True          False

   START YEAR_START YEAR
0                 False
1                 False
2                 False
3                 False
```

Fig. 26d. Transformación de características categóricas.

Finalizado el proceso de preprocesamiento, se empleó la librería FLAML con el objetivo de identificar los algoritmos de ML más adecuados para la predicción del tiempo de construcción y del costo del proyecto en este conjunto de datos.

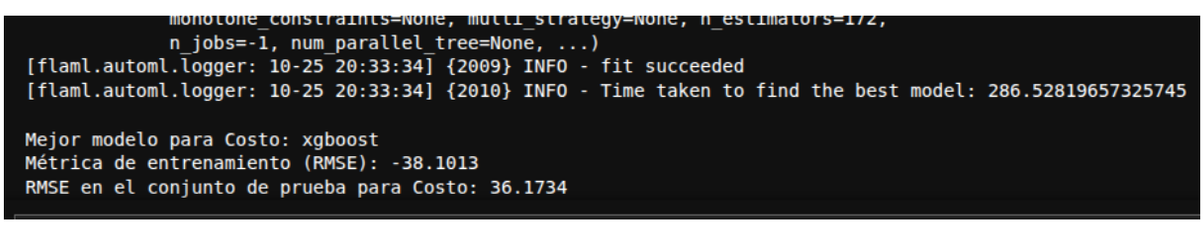
Como resultado de esta optimización automática de modelos, FLAML determinó que el algoritmo ***Stochastic Gradient Descent*** **(SGD)** es el más apropiado para la estimación del tiempo de construcción, obteniendo el mejor desempeño en las métricas evaluadas (Fig. 27).



```
[flaml.automl.logger: 10-25 19:45:38] {2724} INFO - retrain sgd for 0.0s
[flaml.automl.logger: 10-25 19:45:38] {2727} INFO - retrained model: SGDRegressor(alpha=0.00010223665070559743, learning_rate='optimal',
              loss='epsilon_insensitive', penalty='l1', tol=0.0001)
[flaml.automl.logger: 10-25 19:45:38] {2009} INFO - fit succeeded
[flaml.automl.logger: 10-25 19:45:38] {2010} INFO - Time taken to find the best model: 136.19817352294922

Mejor modelo para Tiempo: sgd
Métrica de entrenamiento (RMSE): -1.0189
RMSE en el conjunto de prueba para Tiempo: 1.1452
```

Fig. 27. Salida de FLAML para estimar el tiempo.

De manera análoga, para identificar el modelo más adecuado en la predicción del costo total del proyecto, se aplicó nuevamente FLAML, el cual recomendó el algoritmo con el mejor desempeño para esta tarea (Fig. 28).



```
              monotone_constraints=None, multi_strategy=None, n_estimators=172,
              n_jobs=-1, num_parallel_tree=None, ...)
[flaml.automl.logger: 10-25 20:33:34] {2009} INFO - fit succeeded
[flaml.automl.logger: 10-25 20:33:34] {2010} INFO - Time taken to find the best model: 286.52819657325745

Mejor modelo para Costo: xgboost
Métrica de entrenamiento (RMSE): -38.1013
RMSE en el conjunto de prueba para Costo: 36.1734
```

Fig. 28. Salida de FLAML para estimar el costo.

A continuación, se muestra el resultado que se obtuvo durante el entrenamiento y prueba para estimar el tiempo con el modelo SGD (Fig. 29).

Como se observa en sla Figs. 29 a y29b, la evaluación del modelo SGD evidencia un rendimiento moderado: el MAE = 1.2879 resulta considerablemente menor que el RMSE = 1.7848, lo que indica la presencia de algunos errores elevados y, por ende, posibles *outliers*. Por su parte, el coeficiente de determinación $R^2 = 0.3892$ (aprox. 38.92 %) sugiere que el modelo explica una fracción limitada de la variabilidad en la duración de los proyectos, dejando alrededor del 61 % de la variación sin capturar.

Frente a este desempeño, se decidió explorar modelos no lineales de mayor complejidad, tales como **XGBoost** y **LightGBM**, los cuales fueron evaluados también con FLAML, con el objetivo de incrementar el valor de R^2 por encima del 0.60 o incluso 0.70 (Fig. 30a y 30b).

Entre los tres modelos evaluados (SGD, LGBM y XGBoost), la opción más adecuada para predecir el tiempo de finalización es SGD. Este modelo obtiene el mejor desempeño en el error RMSE (1.7848) —indicador sensible a la presencia de *outliers*— y se posiciona segundo en MAE (1.2879). Además, presenta el mayor poder explicativo, con un $R^2 = 0.3892$, lo que lo convierte en la alternativa más sólida dentro del conjunto de algoritmos analizados (Fig. 31).

Para la estimación del costo del proyecto, se empleó el algoritmo XGBoost, recomendado por el proceso de selección automática de modelos (Fig. 32).

Como se observa en las métricas de la Fig. 32, el modelo alcanzó un $R^2 = 0.9249$ (92.49 %), lo que constituye un desempeño sobresaliente. Este nivel de ajuste indica que XGBoost es capaz de explicar prácticamente toda la variabilidad de los costos de construcción a partir de las características del proyecto y de las variables económicas consideradas. En consecuencia, se trata de un modelo con alta capacidad predictiva y robustez en el análisis.

```
======================================================
  ENTRENANDO MODELO SGD PARA ESTIMACIÓN DE TIEMPO (V7)
======================================================
Modelo SGD entrenado exitosamente.

--- METRICAS DE EVALUACIÓN (CONJUNTO DE PRUEBA) ---
RMSE (Error Cuadrático Medio Raíz): 1.7848
MAE (Error Absoluto Medio):         1.2879
R2 (Coeficiente de Determinación): 0.3892
------------------------------------------------------
```

Fig. 29a. Métricas para SGD.

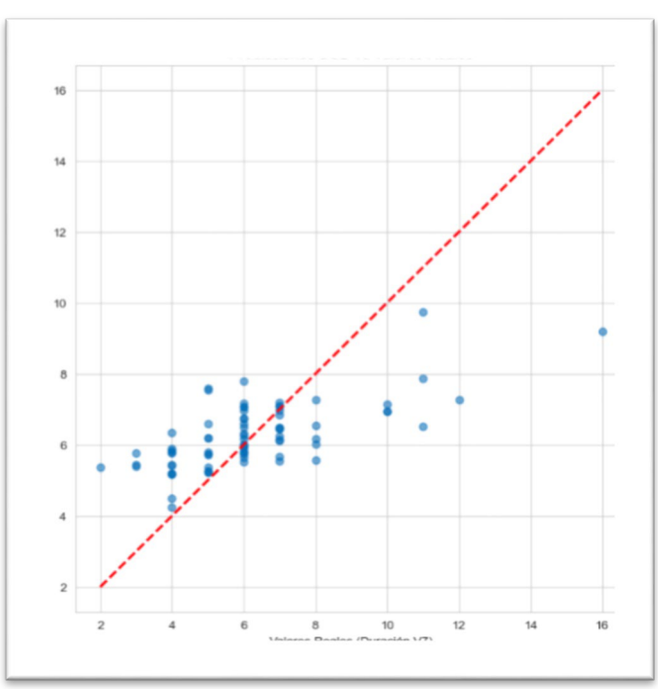


Fig. 29b– Predicción de SGD.

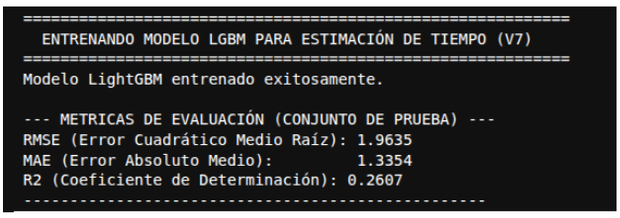


Fig. 30a. Métricas del modelo LGBM.

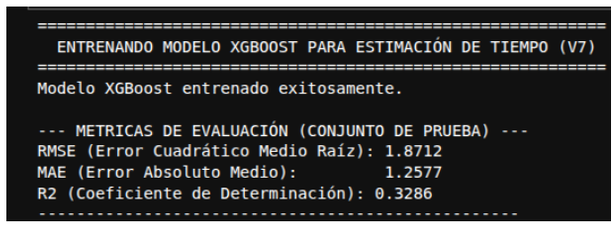


Fig. 30b. Métricas del modelo XGBoost.

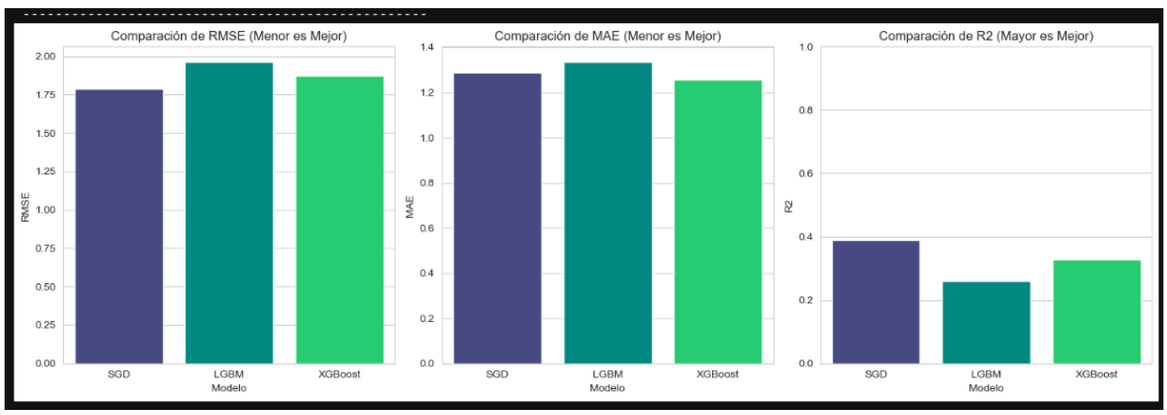


Fig. 31. Comparación de modelos para predecir el tiempo.

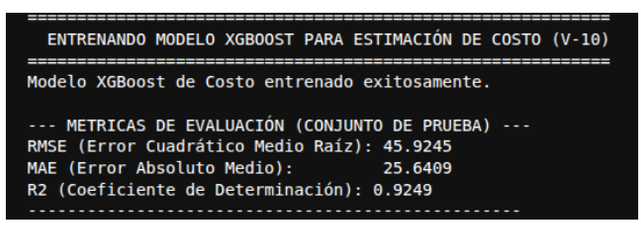


Fig. 32. Métricas de XGBoost.

3.2. Observaciones de los modelos aplicados

En función del alcance definido para este estudio, la evaluación de los modelos se realizó mediante validación cruzada estándar y partición entrenamiento/prueba, según el conjunto de datos analizado. No se aplicaron esquemas de validación cruzada anidada, análisis formal de incertidumbre ni técnicas de explicabilidad de modelos, los cuales se consideran extensiones metodológicas relevantes para etapas futuras del trabajo.

Se entrenaron y evaluaron múltiples modelos de regresión utilizando bibliotecas como scikit-learn, pandas y matplotlib. La validación del rendimiento se realizó sobre el conjunto de prueba, empleando métricas estándar como el coeficiente de determinación (R^2).

Los resultados iniciales obtenidos para la predicción de los días de retraso reflejaron un desempeño deficiente ($R^2 = -0.07$), lo que evidencia que el modelo no lograba capturar adecuadamente la variabilidad del fenómeno. Esta situación motivó la revisión y ajuste de las variables sintéticas para incrementar su representatividad.

Estos hallazgos subrayan la importancia de formular hipótesis sólidas y validadas antes de intervenir los datos, con el fin de prevenir el sesgo de adaptar los datos a la hipótesis, cuando en realidad debe ocurrir lo contrario.

3.3. Implementación y monitoreo

Si bien este trabajo no contempla una implementación en producción, en etapas futuras se prevé la integración de los modelos desarrollados en aplicaciones web o sistemas de gestión de proyectos, con el fin de posibilitar su uso operativo por parte de los actores del sector. Asimismo, se plantea la comparación del desempeño de dichos modelos utilizando datos reales, contrastándolos con métodos tradicionales como CPM, PERT y la simulación Montecarlo. Finalmente, se destaca la importancia del monitoreo, mantenimiento y actualización periódica de los modelos predictivos, dado que los patrones de los datos pueden evolucionar con el tiempo y afectar la estabilidad y precisión de las predicciones.

4. Conclusiones

El estudio demuestra que los modelos de IA, en particular los basados en aprendizaje automático como *Random Forest*, Extra Trees y XGBoost, ofrecen un desempeño predictivo altamente confiable para la estimación de costos de construcción, alcanzando niveles de precisión superiores al 90 %. Estos resultados confirman el potencial de la IA como herramienta estratégica de apoyo a la gestión y planificación de obras en contextos caracterizados por alta incertidumbre.

No obstante, la precisión en la predicción del tiempo de finalización depende en gran medida de la disponibilidad y calidad de los datos de entrada. La limitada cantidad de conjuntos de datos reales en la provincia de Misiones constituyó la principal restricción del estudio, lo que pone de manifiesto la necesidad urgente de desarrollar una base de datos estructurada, abierta y actualizable sobre proyectos de construcción a nivel provincial.

En este marco, se recomienda avanzar hacia la integración de fuentes heterogéneas de información (técnicas, económicas y geoespaciales) en una plataforma interoperable, idealmente dentro del ecosistema IDE Misiones, a fin de facilitar la generación de modelos predictivos robustos y escalables. Asimismo, se proyecta la futura incorporación de técnicas de aprendizaje profundo y modelos híbridos IA–BIM, que permitan abordar dimensiones adicionales como sostenibilidad, eficiencia energética y gestión de riesgosAdemás, se prevé la incorporación de esquemas de validación cruzada anidada, análisis de incertidumbre e interpretabilidad de los modelos predictivos, con el objetivo de fortalecer la robustez, transparencia y aplicabilidad operativa de los resultados obtenidos.

En síntesis, este trabajo constituye un aporte metodológico relevante para la digitalización del sector de la construcción en entornos regionales, evidenciando que la IA puede transformar la forma en que se planifican, ejecutan y controlan los proyectos, al ofrecer estimaciones más precisas, ágiles y respaldadas por datos.

Referencias

[1] Djatnika, Suntana Sukma, Witjaksana, Budi, Oetomo, Wateno, y Bambang, W., «Risk Analysis on the Construction Project of the Basement Building under Jalan Pemuda-Yos Sudarso, Surabaya», *J. Phys. Conf. Ser*., vol. 1364, 2019, doi: 10.1088/1742-6596/1364/1/012073.

[2] Ullah, K, Abdullah, A. H., Nagapan, S., Suhoo, S., y Khan, M. S., «Theoretical framework of the causes of construction time and cost overruns»,*IOP Conf. Ser. Mater. Sci. Eng*., vol. 271, 2017, doi: 10.1088/1757-899X/271/1/012032.

[3] Anastasia Stepanets, «Método de la ruta crítica en la gestión de proyectos», https://blog.ganttpro.com/. [En línea]. Disponible en: https://blog.ganttpro.com/es/metodo-de-la-ruta-critica-en-la-administracion-de-proyectos/. Accedido el: 28/10/2025.

[4] Team Asana, «El diagrama de PERT: qué es y cómo crearlo (incluye ejemplos)». [En línea]. Disponible en: https://asana.com/. Accedido el: 28/10/2025.

[5] Sonia Coma, «Método de simulación Montecarlo: Qué es y cómo adoptarlo en tu empresa». [En línea]. Disponible en: https://asana.com/. Accedido el: 28/10/2025.

[6] Setiawan, A, Fadjar, A., y Labombang, M., «Scheduling the Construction of Low-Income Community Houses in Palu City Using the Probabilistic Duration Method», *IOP Conf. Ser. Earth Environ. Sci.* , vol. 1355, 2024, doi: 10.1088/1755-1315/1355/1/012013.

[7] Mirindi, Derrick, Mirindi, Frederic, Bezabih, Tajebe, Sinkhonde, David, y Kiarie, Winfred, «Review: The Role of Artificial Intelligence in Building Information Modeling», *ACM - SIMIS-CPR*, 2025, doi: 10.1145/3716489.3728433.

[8] Nihad Hassan, «AI vs. machine learning vs. deep learning: Key differences», https://www.techtarget.com/. [En línea]. Disponible en: https://www.techtarget.com/searchenterpriseai/tip/AI-vs-machine-learning-vs-deep-learning-Key-differences/?utm_source=chatgpt.com. Accedido el: 28/10/2025.

[9] I. H. Sarker, «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions», *SN Comput. Sci.*, vol. 2, 2021, doi: 10.1007/s42979-021-00815-1.

[10] Demiss, Belachew A. y Elsaigh, Walied A., «Application of novel hybrid deep learning architectures combining Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN): construction duration estimates prediction considering preconstruction uncertainties», *Eng. Res. Express*, vol. 6, 2024, doi: 10.1088/2631-8695/ad6ca7.

[11] Soreti M Liben, Demiss A Belachew, y Walied A Elsaigh, «Comparing advanced and traditional machine learning algorithms for construction duration prediction: a case study of Addis Ababa's public sector», *Eng. Res. Express*, vol. 6, 2024, doi: 10.1088/2631-8695/ad979f.

[12] Mohammad Savargiv, Behrooz Masoumi, y Mohammad Reza Keyvanpour, «A New Random Forest Algorithm Based on Learning», *Comput. Intell. Neurosci*, 2021, doi: 10.1155/2021/5572781.

[13] Zi-Yi Yang, Zhao-Feng Ye, Chang-Yu Hsieh, y Yi-Jia Xiao, «SPLDE XTRA T REES : R OBUST MACHINE LEARNING APPROACH FOR PREDICTING KINASE INHIBITOR RESISTANCE», *arXiv preprint*, 2021.

[14] Cynthia Lokker, Wael Abdelkader, y Elham Bagheri, «Boosting efficiency in a clinical literature surveillance system with LightGBM», *PLOS Digit. Health*, 2024, doi: /10.1371/%20journal.pdig.0000299.

[15] Chi Wang, Qingyun Wu, Markus Weimer, y Erkang Zhu, «FLAML: A F AST AND L IGHTWEIGHT AUTO ML LIBRARY», *Proc. 4 Th MLSys Conf.*, 2021.

[16] Arunn Thevapalan, «The Machine Learning Workflow Explained (and How You Can Practice It Now)». [En línea]. Disponible en: https://towardsdatascience.com/. Accedido el: 28/10/2025.

[17] Deepakcode07, «Construction-Project-Data-Generator». https://github.com/. [En línea]. Disponible en: https://github.com/Deepakcode07/Construction-Project-Data-Generator?utm_source=chatgpt.com. Accedido el: 28/10/2025.

[18] «JohnVans123/ProjectManagement». [En línea]. Disponible en: https://huggingface.co/datasets/JohnVans123/ProjectManagement?utm_source=chatgpt.com. Accedido el: 28/10/2025.

[19] Mohammad Rafiei, «Residential Building». [En línea]. Disponible en: https://archive.ics.uci.edu/dataset/437/residential%2Bbuilding%2Bdata%2Bset?utm_source=chatgpt.com. Accedido el: 28/10/2025.

[20] Google, *NotebookLM*. [En línea]. Disponible en: https://notebooklm.google/. Accedido el: 28/10/2025.