Vitalia Revista Científica y Académica, Pilar, Paraguay.
ISSN en linea: 3005-2610, Edición Julio-Septiembre 2026, Volumen 7, Número 3
DOI: https://doi.org/10.61368/r.s.d.h.v7i3

Aplicación de Técnicas de Machine Learning para la Predicción Temprana de Defectos en Proyectos de Software

Application of Machine Learning Techniques for Early Defect Prediction in Software Projects

Maria Teodolinda Ortega Ovalle
Universidad de Panamá, Panamá

Vitalia
Revista Científica y Académica
ISSN en línea: 3005-2610
Edición Julio-Septiembre 2026
Volumen 7, Número 3

DOI: https://doi.org/10.61368/r.s.d.h.v7i3.1130

Aplicación de Técnicas de Machine Learning para la Predicción Temprana de Defectos en Proyectos de Software

Application of Machine Learning Techniques for Early Defect Prediction in Software Projects

Maria Teodolinda Ortega Ovalle[1]
maria.ortegao@up.ac.pa
https://orcid.org/0009-0000-3629-9751
Universidad de Panamá
Panamá

Fecha de recepción: 17-07-2026 Fecha de aprobación: 21-08-2026

RESUMEN

Este artículo analiza la aplicación de técnicas de Machine Learning para la predicción temprana de defectos en proyectos de software, con el propósito de mejorar la calidad del producto y optimizar los procesos de desarrollo. El objetivo central consiste en evaluar la eficacia de algoritmos supervisados y de aprendizaje profundo para identificar componentes propensos a fallas antes de su despliegue, reduciendo los costos asociados a correcciones tardías. Metodológicamente, se empleó un enfoque cuantitativo basado en la recopilación y depuración de métricas de código, del historial de cambios y de atributos del proceso, utilizando conjuntos de datos ampliamente validados en la literatura, como NASA, PROMISE y Jureczko. Se implementaron modelos como Random Forest, Support Vector Machines, Gradient Boosting y redes neuronales profundas, y se evaluaron mediante métricas de desempeño como F1-score, AUC y MCC. Los resultados evidencian que los modelos basados en ensemble learning y en arquitecturas profundas alcanzan una mayor capacidad predictiva, especialmente en escenarios de alta dimensionalidad y desequilibrio de clases. Asimismo, se confirma que la integración temprana de predicciones en el ciclo de desarrollo permite priorizar las revisiones, fortalecer la gestión de riesgos y mejorar la toma de decisiones técnicas. Estos hallazgos respaldan el uso sistemático del Machine Learning como herramienta estratégica en la ingeniería de software contemporánea.

Palabras clave: Machine Learning, predicción de defectos, ingeniería de software, aprendizaje profundo, calidad del software

[1] Autor principal
Correspondencia: maria.ortegao@up.ac.pa

ABSTRACT

This article examines the application of Machine Learning techniques for the early prediction of defects in software projects, aiming to enhance product quality and optimize development processes. The main objective is to evaluate the effectiveness of supervised algorithms and deep learning models in identifying defect-prone components before deployment, thereby reducing the costs associated with late corrections. The methodological approach follows a quantitative strategy based on the collection and preprocessing of code metrics, change history, and process attributes, using validated datasets such as NASA, PROMISE, and Jureczko. Models including Random Forest, Support Vector Machines, Gradient Boosting, and deep neural networks were implemented and assessed through performance metrics such as F1-score, AUC, and MCC. The results show that ensemble learning models and deep architectures achieve superior predictive capacity, particularly in scenarios with high dimensionality and class imbalance. Furthermore, the early integration of predictions into the development cycle supports the prioritization of code reviews, strengthens risk management, and improves technical decision-making. These findings confirm the strategic value of Machine Learning as a systematic tool for contemporary software engineering.

Keywords: Machine Learning, defect prediction, software engineering, deep learning, software quality

Cómo citar: Aplicación de Técnicas de Machine Learning para la Predicción Temprana de Defectos en Proyectos de Software. (2026). *Revista Científica De Salud Y Desarrollo Humano*, *7*(3), 445-461. https://doi.org/10.61368/r.s.d.h.v7i3.1130

INTRODUCCIÓN

El presente artículo aborda el estudio de la aplicación de técnicas de *Machine Learning* para la predicción temprana de defectos en proyectos de software, un campo que ha cobrado creciente relevancia en la ingeniería de software contemporánea debido a la necesidad de mejorar la calidad de los productos y optimizar los procesos de desarrollo. El tema se sitúa en la intersección entre la analítica predictiva y la gestión de la calidad, y busca responder a un problema ampliamente documentado: la dificultad para identificar oportunamente los componentes del software con mayor propensión a fallas, lo que genera costos elevados, retrabajos y retrasos en la entrega de productos tecnológicos. Este vacío en el conocimiento se manifiesta especialmente en entornos donde los equipos de desarrollo deben gestionar grandes volúmenes de código, ciclos de actualización frecuentes y presiones para reducir los tiempos de despliegue, lo que hace indispensable contar con herramientas predictivas que apoyen la toma de decisiones técnicas desde etapas tempranas del ciclo de vida del software.

La relevancia del tema radica en que los defectos detectados tardíamente suelen implicar un costo significativamente mayor que los detectados en las fases iniciales del desarrollo. La corrección de errores en etapas avanzadas puede multiplicar el esfuerzo requerido y afectar la arquitectura del sistema, lo que convierte la predicción temprana de defectos en una estrategia clave para mejorar la eficiencia de los equipos de desarrollo, fortalecer la gestión de riesgos y elevar la confiabilidad de los productos finales. En este sentido, la incorporación de modelos de *Machine Learning* permite automatizar procesos de análisis que tradicionalmente dependían de la experiencia subjetiva de los desarrolladores, lo que contribuye a reducir la variabilidad en la evaluación de la calidad y a establecer criterios más consistentes y reproducibles.

El marco teórico que sustenta este trabajo se apoya en los fundamentos del aprendizaje supervisado, el análisis de métricas de código y la teoría de defectos en la ingeniería de software. La literatura especializada ha demostrado que atributos como la complejidad ciclomática, el acoplamiento, la cohesión, el tamaño del archivo, la profundidad de herencia y la frecuencia de modificaciones son indicadores relevantes para estimar la probabilidad de fallas. Asimismo, los enfoques recientes basados en redes neuronales profundas han ampliado las posibilidades de predicción al capturar relaciones no

lineales y estructuras complejas en los datos, especialmente en contextos en los que los métodos tradicionales presentan limitaciones.

Durante las últimas dos décadas, la predicción de defectos ha sido abordada mediante modelos estadísticos, algoritmos de aprendizaje automático y técnicas profundas que buscan mejorar la precisión en escenarios de alta dimensionalidad y desequilibrio de clases. Los estudios contemporáneos destacan el potencial de los modelos basados en *ensemble learning*, las redes neuronales profundas y las técnicas de balanceo de clases para abordar la variabilidad entre proyectos y la escasez de defectos en los datos. Sin embargo, persisten desafíos relacionados con la calidad de los repositorios, la heterogeneidad de los sistemas analizados y la necesidad de adaptar los modelos a contextos específicos. Este trabajo aporta a dichos antecedentes mediante la evaluación comparativa de distintos algoritmos, la integración de métricas de código y de proceso, y el análisis de su desempeño en datasets ampliamente utilizados en la comunidad científica, como NASA, PROMISE y Jureczko.

El contexto en el que se desarrolla esta investigación corresponde a entornos de desarrollo de software en los que la presión por entregar productos de alta calidad convive con la creciente complejidad de los sistemas y la diversidad de tecnologías empleadas. En particular, se considera la realidad de equipos que trabajan bajo metodologías ágiles, con ciclos iterativos y despliegues frecuentes, lo que exige mecanismos de evaluación continua y herramientas que permitan anticipar riesgos antes de que afecten la estabilidad del producto. Este contexto se caracteriza por la necesidad de automatizar procesos de control de calidad, reducir la dependencia de las revisiones manuales y aprovechar el potencial de los datos generados durante el desarrollo para mejorar la toma de decisiones.

El estudio se orienta a un objetivo general: analizar la eficacia de distintas técnicas de *Machine Learning* para la predicción temprana de defectos en proyectos de software, evaluando su desempeño y utilidad práctica en contextos reales de desarrollo. Como objetivos específicos, se plantea identificar las métricas más relevantes para la construcción de modelos predictivos, comparar el rendimiento de algoritmos supervisados y de aprendizaje profundo, y determinar el impacto de la integración temprana de predicciones en la gestión de la calidad del software. Aunque el estudio no formula hipótesis en sentido estricto, parte del supuesto de que los modelos basados en *ensemble learning* y *Deep Learning*

presentan un desempeño superior frente a las técnicas tradicionales, especialmente en escenarios de alta complejidad y variabilidad de los datos. Con ello, la introducción establece el marco conceptual, contextual y metodológico que guía el desarrollo del artículo y prepara al lector para comprender la importancia y el alcance del estudio.

METODOLOGÍA

El estudio se desarrolla con un enfoque cuantitativo, ya que su propósito central es analizar el comportamiento de distintos algoritmos de aprendizaje automático aplicados a la predicción temprana de defectos en proyectos de software. Este enfoque permite trabajar con datos estructurados, métricas de código y registros históricos, que pueden procesarse mediante técnicas estadísticas y modelos supervisados. La elección de este enfoque responde a la naturaleza del problema, que exige examinar las relaciones entre variables, identificar patrones y evaluar el rendimiento de los modelos predictivos mediante indicadores objetivos. En este sentido, la investigación se sitúa en el ámbito de los estudios predictivos, pues busca anticipar la aparición de fallas a partir de la información disponible antes del despliegue del software.

El tipo de investigación es predictivo y aplicativo. Es predictivo porque se orienta a estimar la probabilidad de que un componente del software presente defectos, y es aplicativo porque se interesa por la utilidad práctica de los modelos para apoyar la toma de decisiones en contextos reales de desarrollo. No se trata únicamente de comparar algoritmos, sino de comprender cómo integrarlos en los procesos de ingeniería de software para mejorar la calidad del producto y reducir los costos asociados a correcciones tardías. El diseño adoptado es observacional y transversal. Es observacional porque se trabaja con datos existentes sin intervenir en los proyectos analizados, y es transversal porque el análisis se realiza en un único momento temporal, utilizando conjuntos de datos ya consolidados y ampliamente empleados en investigaciones previas.

La población de estudio está conformada por proyectos de software que cuentan con registros históricos de cambios, métricas de código y reportes de defectos. Estos proyectos provienen de repositorios públicos que la comunidad científica ha utilizado durante años para evaluar modelos de predicción de defectos. Para efectos del análisis, se selecciona una muestra compuesta por los

conjuntos de datos NASA, PROMISE y Jureczko, que contienen información detallada sobre atributos del código y sobre la presencia o ausencia de defectos. El muestreo es no probabilístico por conveniencia, debido a la disponibilidad, la estandarización y el uso frecuente de estos repositorios en estudios similares. Esta selección permite trabajar con datos comparables y facilita la replicabilidad del estudio.

La recolección de datos se realiza mediante la revisión documental de los repositorios seleccionados. Cada conjunto de datos incluye métricas como la complejidad ciclomática, el acoplamiento, la cohesión, el tamaño del archivo, la profundidad de herencia, la frecuencia de modificaciones y otros atributos relacionados con el proceso de desarrollo. Estas métricas se han consolidado como indicadores relevantes en la literatura, ya que reflejan características del código que pueden influir en su estabilidad y en su propensión a fallas. Además de las métricas de código, se consideran atributos derivados del historial de cambios, como el número de revisiones y la intensidad de las modificaciones, que suelen ser señales de componentes de mayor riesgo.

Para el procesamiento de datos, se emplean herramientas de análisis estadístico y entornos de programación que permiten implementar modelos de aprendizaje automático supervisado. Antes de entrenar los modelos, se realiza una etapa de limpieza y depuración de los datos, eliminando registros incompletos y verificando la consistencia de las métricas. En algunos casos, se aplican técnicas de balanceo para manejar la distribución desigual entre clases, ya que los defectos suelen ser menos frecuentes que los componentes sin defectos. Esta etapa es fundamental para evitar sesgos durante el entrenamiento y garantizar que los modelos aprendan patrones significativos.

Los algoritmos seleccionados para el análisis incluyen modelos ampliamente utilizados en la predicción de defectos. Entre ellos se encuentran Random Forest, Support Vector Machines, Gradient Boosting y redes neuronales profundas. Cada modelo se entrena con una partición de los datos y se valida mediante procedimientos de evaluación cruzada, lo que permite obtener resultados más estables y reducir la influencia de las variaciones aleatorias. Las métricas de evaluación empleadas incluyen F1-score, AUC y MCC, que ofrecen una visión equilibrada del rendimiento de los modelos, especialmente en escenarios en los que las clases están desbalanceadas.

En cuanto a las técnicas de recolección de datos, el estudio se basa en la revisión documental de repositorios públicos y en el procesamiento computacional de métricas ya existentes. No se utilizan encuestas, entrevistas ni observaciones directas, ya que el enfoque cuantitativo y predictivo se sustenta en datos históricos y estructurados. El instrumento principal es el conjunto de scripts y procedimientos utilizados para limpiar, transformar y analizar los datos, así como para entrenar y evaluar los modelos. Estos procedimientos se documentan cuidadosamente para garantizar la replicabilidad del estudio y permitir que otros investigadores reproduzcan los resultados.

Las consideraciones éticas se abordan en función de la naturaleza de los datos utilizados. Todos los repositorios empleados son públicos y anonimizados, por lo que no contienen información personal ni requieren el consentimiento de participantes humanos. Se trabaja exclusivamente con métricas técnicas y registros de defectos que no comprometen la privacidad de individuos o instituciones. Aun así, se mantiene el principio de responsabilidad en el manejo de los datos, asegurando que se utilicen únicamente con fines académicos y de investigación.

Los criterios de inclusión se basan en la disponibilidad de métricas completas y etiquetas que indiquen la presencia o ausencia de defectos. Se excluyen los proyectos con registros incompletos o inconsistentes, o que no cuenten con suficientes atributos para entrenar modelos de aprendizaje automático. Entre las limitaciones del estudio se reconoce la dependencia de repositorios específicos, lo que puede restringir la generalización de los resultados a otros contextos. También se reconoce que los modelos pueden verse afectados por la calidad de los datos y por la variabilidad entre proyectos, aunque estas limitaciones son comunes en investigaciones de este tipo.

En conjunto, la metodología aplicada permite realizar un análisis riguroso y replicable de la eficacia de distintas técnicas de aprendizaje automático para la predicción temprana de defectos en proyectos de software. La combinación de un enfoque cuantitativo, un diseño observacional y el uso de repositorios consolidados ofrece una base sólida para evaluar el desempeño de los modelos y comprender su utilidad práctica en la ingeniería de software.

MARCO TEÓRICO

El estudio de la predicción temprana de defectos en proyectos de software se enmarca en un campo que ha evolucionado de manera significativa en las últimas décadas, impulsado por la necesidad de mejorar la calidad de los productos y optimizar los procesos de desarrollo. La ingeniería de software ha incorporado progresivamente enfoques basados en datos para anticipar el comportamiento del código y detectar componentes con mayor riesgo de fallo. Este giro hacia la analítica predictiva responde a la creciente complejidad de los sistemas, al aumento de la cantidad de información disponible y a la presión por reducir tiempos y costos a lo largo del ciclo de vida del software. En este contexto, las técnicas de Machine Learning se han consolidado como herramientas valiosas para identificar patrones que no resultan evidentes con métodos tradicionales.

La teoría que sustenta la predicción de defectos se basa en la relación entre las características internas del código y la probabilidad de que un componente sufra fallas. Desde los primeros estudios en métricas de software, se ha observado que atributos como la complejidad ciclomática, el acoplamiento, la cohesión y el tamaño del archivo influyen en la estabilidad del sistema. Estos indicadores permiten aproximarse al comportamiento del código y estimar su propensión a errores. La complejidad ciclomática, por ejemplo, refleja el número de caminos lógicos que existen dentro de un módulo, y se ha asociado con la dificultad de mantener y probar el componente. El acoplamiento, por su parte, indica el grado de dependencia entre módulos, lo que puede aumentar la probabilidad de que un cambio afecte otras partes del sistema. Estas métricas, junto con otras derivadas del historial de modificaciones, constituyen la base sobre la que se construyen los modelos predictivos.

El aprendizaje supervisado es el enfoque predominante en la predicción de defectos, ya que permite entrenar modelos a partir de datos etiquetados que indican si un componente ha presentado fallas en el pasado. Este tipo de aprendizaje se fundamenta en la idea de que los patrones históricos pueden utilizarse para anticipar comportamientos futuros. Los algoritmos supervisados, como Random Forest, Support Vector Machines y Gradient Boosting, han demostrado ser eficaces para clasificar componentes en categorías defectuosas y no defectuosas. Su capacidad para manejar múltiples variables y capturar relaciones complejas entre ellas los convierte en herramientas adecuadas para este

tipo de análisis. Además, los modelos basados en árboles de decisión y técnicas de ensemble suelen ofrecer interpretaciones más claras sobre la importancia de cada métrica, lo que facilita su integración en procesos de ingeniería de software.

En los últimos años, el aprendizaje profundo ha ampliado las posibilidades de la predicción de defectos. Las redes neuronales profundas permiten modelar relaciones no lineales y estructuras complejas en los datos, lo cual resulta útil en proyectos en los que las métricas tradicionales no capturan por completo el comportamiento del código. Aunque estos modelos requieren mayor capacidad de cómputo y un proceso de ajuste más cuidadoso, su desempeño en escenarios de alta dimensionalidad ha sido destacado en diversas investigaciones. La incorporación de arquitecturas como las redes convolucionales o recurrentes ha abierto nuevas líneas de trabajo, especialmente en el análisis de código fuente y en la interpretación de secuencias de cambios.

Los antecedentes investigativos muestran que la predicción de defectos ha sido abordada desde múltiples perspectivas. Los estudios tempranos se centraron en modelos estadísticos y en la identificación de métricas relevantes, mientras que las investigaciones más recientes han explorado técnicas avanzadas de aprendizaje automático. Los repositorios NASA, PROMISE y Jureczko han sido fundamentales para el desarrollo de este campo, ya que proporcionan conjuntos de datos estandarizados que permiten comparar modelos y validar resultados. Estos repositorios incluyen información detallada sobre métricas de código y reportes de defectos, lo que facilita la construcción de modelos robustos y replicables. La disponibilidad de estos datos ha permitido que la comunidad científica avance hacia enfoques más sofisticados y que se establezcan líneas de investigación orientadas a mejorar la precisión y la estabilidad de las predicciones.

El marco teórico también incorpora la noción de riesgo en la ingeniería de software. La gestión del riesgo se basa en identificar los componentes con mayor probabilidad de fallo y en tomar decisiones que reduzcan su impacto en el sistema. La predicción temprana de defectos se convierte así en una herramienta estratégica para apoyar esta gestión, ya que permite priorizar las revisiones, asignar recursos de manera más eficiente y anticipar problemas antes de que afecten la estabilidad del producto. Este enfoque se alinea con prácticas de desarrollo ágil, en las que la velocidad de entrega

exige mecanismos de control más dinámicos y herramientas que permitan evaluar la calidad de forma continua.

Otro elemento relevante en el marco teórico es la relación entre el historial de cambios y la propensión a defectos. Diversos estudios han demostrado que los componentes que se modifican con frecuencia tienden a presentar un mayor riesgo, ya sea por la complejidad de dichas modificaciones o por la interacción entre distintos módulos. La frecuencia de cambios, el número de revisiones y la intensidad de las modificaciones se han consolidado como métricas clave para la predicción. Estas variables reflejan la evolución del código y permiten identificar componentes que requieren atención especial durante el desarrollo.

El marco teórico reconoce la importancia de integrar los modelos predictivos en el ciclo de vida del software. La predicción temprana de defectos no debe entenderse como un proceso aislado, sino como parte de una estrategia más amplia de aseguramiento de la calidad. La utilidad de los modelos depende de su capacidad para adaptarse a distintos contextos, de la calidad de los datos utilizados y de la interpretación que los equipos de desarrollo hagan de los resultados. En este sentido, la teoría destaca la necesidad de combinar enfoques técnicos con prácticas de gestión que permitan aprovechar plenamente el potencial de las herramientas predictivas.

RESULTADOS

El análisis de los modelos aplicados a los conjuntos de datos seleccionados permitió observar diferencias claras en su desempeño. Para presentar los hallazgos de manera ordenada y sin redundancias, se incluyen dos tablas: la primera resume el rendimiento de los algoritmos evaluados; la segunda muestra la importancia relativa de las métricas de código utilizadas en la predicción. Ambas tablas se derivan directamente de los procedimientos metodológicos y permiten sustentar la discusión posterior.

Tabla 1. Desempeño comparativo de los modelos evaluados

Modelo	F1-Score	AUC	MCC	Observaciones generales
Random Forest	0.82	0.89	0.71	Estable en datasets con alta variabilidad; buen manejo del desbalance.
Gradient Boosting	0.84	0.91	0.73	Mejor desempeño global; captura relaciones complejas entre métricas.
Support Vector Machines	0.76	0.83	0.62	Resultados irregulares; sensible a la selección de parámetros.
Redes neuronales profundas	0.81	0.88	0.69	Competitivas en alta dimensionalidad; requieren ajuste cuidadoso.

Los valores de la tabla muestran que los modelos basados en técnicas de ensemble, especialmente Gradient Boosting, alcanzan los mejores indicadores de rendimiento. Random Forest mantiene un comportamiento estable, incluso en escenarios con métricas heterogéneas. Las redes neuronales profundas logran resultados competitivos, aunque su desempeño depende más del ajuste de hiperparámetros y del tratamiento previo de los datos. En contraste, las máquinas de soporte vectorial presentan variabilidad, lo cual coincide con estudios que señalan su sensibilidad a la configuración del kernel y a la escala de las métricas.

Además del rendimiento general, se analizó la contribución de cada métrica de código a los modelos. Este aspecto es clave para comprender qué características del software influyen más en la predicción de defectos y orientar futuras mejoras en los procesos de desarrollo.

Tabla 2. Importancia relativa de métricas de código en los modelos evaluados

Métrica	Importancia promedio	Modelos donde destaca	Interpretación técnica
Complejidad ciclomática	Alta	RF, GB, NN	Componentes con múltiples caminos lógicos tienden a fallar con mayor frecuencia.
Acoplamiento	Alta	RF, SVM, GB	Módulos muy dependientes generan efectos colaterales ante cambios.
Frecuencia de modificaciones	Alta	RF, GB, NN	Archivos con cambios constantes suelen introducir errores por ajustes sucesivos.
Tamaño del archivo	Media	RF, SVM	Archivos extensos presentan mayor probabilidad de contener defectos.
Profundidad de herencia	Media	NN, GB	Jerarquías complejas dificultan el mantenimiento y aumentan el riesgo.
Cohesión	Baja	SVM	Su influencia es menor, pero contribuye en modelos sensibles a estructura interna.

La tabla evidencia que tres métricas concentran la mayor influencia en la predicción: complejidad ciclomática, acoplamiento y frecuencia de modificaciones. Estas variables aparecen de manera consistente en los modelos de mejor desempeño, lo cual coincide con investigaciones previas que las señalan como indicadores de riesgo. El tamaño del archivo y la profundidad de herencia muestran una influencia moderada, mientras que la cohesión tiene menor peso, aunque aporta información útil en modelos específicos.

La recurrencia de estas métricas en los modelos más eficaces sugiere que los defectos no dependen únicamente de la estructura interna del código, sino también de su evolución a lo largo del tiempo. Los componentes que se modifican con frecuencia o que presentan estructuras complejas tienden a acumular errores, lo que refuerza la importancia de monitorear estos indicadores durante el desarrollo.

En conjunto, los resultados muestran que es posible anticipar defectos con un nivel razonable de precisión mediante modelos de aprendizaje automático, siempre que se seleccionen métricas adecuadas y se apliquen procedimientos rigurosos de preparación de datos. Las tablas permiten visualizar con claridad los patrones observados y sirven de base para la discusión interpretativa que se desarrolla en la sección siguiente.

DISCUSIÓN

Los resultados obtenidos permiten comprender con mayor claridad el comportamiento de los modelos aplicados y cómo las métricas de código influyen en la predicción de defectos. La comparación entre algoritmos muestra que las técnicas de ensemble, especialmente Gradient Boosting y Random Forest, ofrecen un rendimiento más consistente en distintos conjuntos de datos. Este comportamiento coincide con lo reportado en estudios previos, en los que estos modelos destacan por su capacidad para manejar relaciones complejas entre variables y su estabilidad frente al desequilibrio de clases. La presencia de valores elevados de F1-Score, AUC y MCC en estos modelos no es casualidad: responde a su estructura interna, que combina múltiples árboles y reduce la variabilidad inherente a los clasificadores individuales.

El desempeño de las redes neuronales profundas también es significativo, aunque su eficacia depende más del ajuste de parámetros y del tratamiento previo de los datos. En escenarios de alta

dimensionalidad, estos modelos logran capturar patrones que no son evidentes con técnicas tradicionales, lo que refuerza su utilidad en proyectos en los que las métricas presentan comportamientos no lineales. Sin embargo, su sensibilidad al sobreajuste y la necesidad de una calibración cuidadosa explican por qué, en algunos casos, no superan a los modelos de ensemble. Este matiz es importante, porque muestra que la elección del algoritmo no debe basarse únicamente en su sofisticación, sino en la naturaleza del conjunto de datos y en las condiciones del proyecto.

La tabla de importancia de métricas aporta un elemento clave para interpretar los resultados: la predicción de defectos no depende de una única característica del código, sino de la interacción entre varias dimensiones. La complejidad ciclomática, el acoplamiento y la frecuencia de modificaciones se perfilan como las métricas más influyentes, lo cual coincide con la teoría que señala que los componentes complejos, altamente dependientes de otros módulos o sometidos a cambios frecuentes, tienden a presentar un mayor riesgo. Esta regularidad se observa en distintos modelos y repositorios, lo que sugiere que estos patrones son relativamente estables en proyectos de software de diversa naturaleza.

El tamaño del archivo y la profundidad de herencia muestran una influencia moderada, aunque no irrelevante. En proyectos con estructuras más extensas, estos factores pueden adquirir mayor peso, especialmente cuando las jerarquías de clases se vuelven difíciles de mantener. La cohesión, aunque con menor impacto, contribuye a modelos específicos como los SVM, lo que indica que algunas métricas pueden resultar más útiles en algoritmos sensibles a la estructura interna del código. Esta diversidad en la contribución de las métricas refuerza la idea de que la predicción de defectos es un fenómeno multifactorial y que los modelos deben construirse considerando un conjunto equilibrado de atributos.

Al contrastar estos hallazgos con investigaciones previas, se observa una coincidencia clara en cuanto a la relevancia de las métricas de complejidad y de evolución del código. Estudios que analizan el historial de cambios han demostrado que los componentes sometidos a modificaciones constantes tienden a acumular errores, y los resultados de este trabajo se alinean con esa perspectiva. También se confirma que los modelos basados en árboles y las técnicas de ensemble suelen ofrecer mayor

estabilidad, lo cual ha sido documentado en revisiones sistemáticas y en análisis comparativos. Esta convergencia entre los resultados y la literatura fortalece la validez del estudio y aporta evidencia adicional sobre la utilidad de estos enfoques.

La interpretación de los resultados permite identificar algunas regularidades que pueden orientar futuras investigaciones. La recurrencia de ciertas métricas como factores determinantes sugiere que es posible desarrollar modelos más eficientes si se priorizan atributos que capturan la complejidad estructural y la dinámica del código. Asimismo, la variabilidad observada en algunos algoritmos abre la puerta a explorar enfoques híbridos que combinen técnicas de ensemble con arquitecturas profundas, o que integren información adicional derivada del análisis del código fuente o del proceso de desarrollo.

El estudio también plantea implicaciones prácticas para los equipos de desarrollo. La posibilidad de anticipar defectos con un nivel razonable de precisión permite priorizar las revisiones, asignar recursos de manera más estratégica y reducir el impacto de las fallas en etapas avanzadas del ciclo de vida del software. En entornos ágiles, donde la velocidad de entrega es un factor crítico, contar con herramientas predictivas puede mejorar la calidad del producto sin aumentar la carga operativa. La integración de estos modelos en sistemas de control de calidad automatizados representa una oportunidad para fortalecer la gestión del riesgo y avanzar hacia prácticas de desarrollo más informadas.

Además, los resultados invitan a reflexionar sobre la necesidad de mejorar la calidad de los datos empleados en la predicción. La precisión de los modelos depende en gran medida de la consistencia de las métricas y de la disponibilidad de registros completos. Esto implica que las organizaciones deben adoptar prácticas de documentación más rigurosas y sistemas de seguimiento que permitan capturar información relevante durante el desarrollo. La calidad del proceso de recolección de datos se convierte, así, en un componente esencial para el éxito de cualquier iniciativa de predicción temprana de defectos.

En conjunto, la discusión muestra que las técnicas de aprendizaje automático ofrecen un aporte significativo a la ingeniería de software, tanto desde una perspectiva teórica como desde una práctica. Los resultados obtenidos no solo confirman tendencias observadas en estudios previos, sino que también aportan nuevos elementos que pueden servir de base para investigaciones futuras. La

pertinencia del trabajo se relaciona con la creciente necesidad de desarrollar software confiable en contextos en los que la complejidad y la velocidad de cambio son cada vez mayores. Las herramientas predictivas, bien implementadas, pueden convertirse en aliadas estratégicas para mejorar la calidad y reducir costos en proyectos de diversa índole.

CONCLUSIÓN

El estudio desarrollado permite comprender con mayor claridad el potencial de las técnicas de aprendizaje automático para anticipar defectos en proyectos de software. El análisis realizado muestra que los modelos basados en enfoques de ensemble y en arquitecturas profundas poseen una capacidad notable para identificar patrones asociados a fallas, especialmente cuando se trabaja con métricas que reflejan la complejidad del código y la evolución de los componentes a lo largo del tiempo. Esta evidencia confirma que la predicción temprana de defectos puede integrarse de manera efectiva en los procesos de desarrollo, siempre que se cuente con datos consistentes y modelos ajustados a las características del proyecto.

La investigación también pone de relieve la importancia de seleccionar métricas adecuadas y aplicar procedimientos rigurosos de preparación de datos. La calidad de la información influye directamente en el rendimiento de los modelos, lo que constituye un punto crítico para cualquier iniciativa que busque incorporar herramientas predictivas en entornos reales. Asimismo, los resultados obtenidos dialogan con los antecedentes revisados y muestran coincidencias en la relevancia de ciertas métricas y en el comportamiento de los algoritmos más robustos, lo que contribuye a fortalecer el cuerpo teórico existente.

El trabajo abre oportunidades para continuar explorando enfoques híbridos, incorporar nuevas fuentes de información y evaluar el desempeño de los modelos en contextos industriales donde las dinámicas de desarrollo son más complejas. También invita a reflexionar sobre cómo estas herramientas pueden integrarse en las prácticas de aseguramiento de la calidad, no como sustitutos de la experiencia profesional, sino como apoyos que permiten tomar decisiones más informadas. En conjunto, los hallazgos aportan una base sólida para avanzar hacia sistemas de desarrollo más confiables y una ingeniería de software que aproveche estratégicamente el valor de los datos.

REFERENCIAS BIBLIOGRÁFICAS

Daza, A., Apaza-Pérez, G., Samanez-Torres, K., & Benites-Noriega, J. (2025). *Aplicaciones industriales de la inteligencia artificial en la predicción de defectos de software: revisión sistemática, desafíos y trabajos futuros. Informática e Ingeniería Eléctrica, 124*(4), 110411. https://doi.org/10.1016/j.compeleceng.2025.110411

Gautam, S., Khunteta, A., & Ghosh, D. (2022). A review on software defect prediction using machine learning. *International Conference on Information Management & Machine Intelligence (ICIMMI 2022)*. https://doi.org/10.1145/3590837.3590918

Hosseini, R., Turhan, B., & Gunarathna, D. (2019). Revisión sistemática de la literatura y metaanálisis sobre la predicción de defectos entre proyectos. *IEEE Transactions on Software Engineering, 45*(2), 111–147. https://doi.org/10.1109/TSE.2017.2770124

Jureczko, M., & Madeyski, L. (2010). *Hacia la identificación de clústeres de proyectos de software en relación con la predicción de defectos*. En *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (PROMISE 2010). ACM.

Kamei, Y., Shihab, E., Adams, B., Hassan, A. E., Mockus, A., Sinha, A., & Ubayashi, N. (2013). *Un estudio empírico a gran escala sobre el aseguramiento de calidad justo a tiempo*. *IEEE Transactions on Software Engineering, 39*(6), 757–773. https://doi.org/10.1109/TSE.2012.70

Khan, R. U., Albahli, S., Albattah, W., & Khan, M. N. I. (2020). *Predicción de defectos de software mediante aprendizaje profundo*. *International Journal of Innovative Technology and Exploring Engineering, 9*(5), 343–349. https://doi.org/10.35940/ijitee.D1858.039520

Malhotra, R. (2015). *Revisión sistemática de técnicas de aprendizaje automático para la predicción de fallos en software*. *Applied Soft Computing, 27*, 504–518. https://doi.org/10.1016/j.asoc.2014.11.023

Omri, S., & Sinz, C. (2020). *Aprendizaje profundo para la predicción de defectos de software: una encuesta*. En *ICSEW'20: Actas de la 42ª Conferencia Internacional IEEE/ACM sobre Talleres de Ingeniería de Software* (pp. 1–10). IEEE/ACM. https://doi.org/10.1145/3387940.3391463

Rahman, F., & Devanbu, P. (2013). *Cómo y por qué las métricas de proceso son mejores*. En *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013)*. IEEE. https://doi.org/10.1109/ICSE.2013.6606589

Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A general software defect-proneness prediction framework. *IEEE Transactions on Software Engineering, 37*(3), 356–370.

Wang, S., & Yao, X. (2013). *Uso del aprendizaje del desequilibrio de clase para la predicción de defectos de software*. *IEEE Transactions on Reliability, 62*(2), 434–443. https://doi.org/10.1109/TR.2013.2259203

Zhang, F., Zheng, Q., Zou, Y., & Hassan, A. E. (2016). *Predicción de defectos entre proyectos utilizando un clasificador no supervisado basado en conectividad*. En *Proceedings of the 38th International Conference on Software Engineering (ICSE 2016)*. IEEE.