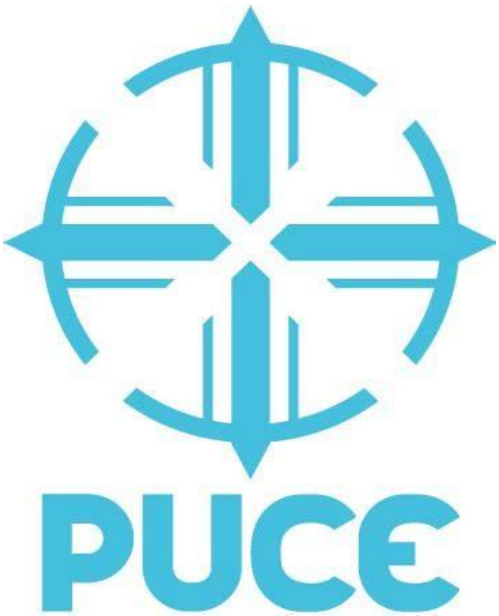
Pontificia Universidad Católica del Ecuador

Facultad De Ingeniería



Tema:

Diseño y evaluación de un modelo basado en aprendizaje automático para estimar calificaciones de tareas de programación mediante la integración de pruebas funcionales y métricas de calidad de código

Autor:

Adrián Estiven Balseca Olivo

Trabajo previo a la obtención del título de Magister en Sistemas de la Información Mención Data Science

Quito – mayo 2026

Título del proyecto

Diseño y evaluación de un modelo basado en aprendizaje automático para estimar calificaciones de tareas de programación mediante la integración de pruebas funcionales y métricas de calidad de código.

Problema de investigación

La evaluación de tareas de programación constituye una actividad central en la formación de estudiantes de carreras tecnológicas, porque permite valorar competencias asociadas al pensamiento lógico, la resolución de problemas, el diseño algorítmico y la aplicación de buenas prácticas de desarrollo de software. No obstante, la revisión manual de estas tareas suele demandar un tiempo considerable, especialmente cuando los docentes deben evaluar múltiples entregas, aplicar rúbricas con varios criterios y mantener consistencia entre estudiantes, paralelos o periodos académicos.

Sin embargo, el problema no se limita únicamente al tiempo requerido para calificar. En programación, una solución no debe evaluarse únicamente por producir una salida correcta; también intervienen aspectos como la claridad del código, la modularidad, la complejidad, la documentación, la mantenibilidad, el manejo de errores y el cumplimiento de buenas prácticas, en este sentido, las herramientas basadas solo en pruebas funcionales pueden ser útiles para verificar si un programa cumple determinados casos de prueba, pero tienden a ofrecer una visión parcial de la calidad de la solución. Por el contrario, una evaluación exclusivamente manual puede introducir variabilidad entre evaluadores, diferencias de criterio o retrasos en la retroalimentación.

En este contexto, surge la necesidad de analizar si los modelos de aprendizaje automático pueden apoyar la estimación de calificaciones de tareas de programación mediante la integración de dos tipos de información: resultados de pruebas funcionales y métricas de calidad de código, sin embargo, para que este enfoque sea válido académicamente, no basta con entrenar un modelo predictivo, pues resulta necesario evaluar si las variables disponibles representan adecuadamente las entregas, si las calificaciones de referencia son consistentes, si el modelo generaliza sobre datos no vistos y si el nivel de error obtenido resulta aceptable para un uso de apoyo docente.

Por tanto, el problema de investigación se centra en determinar la viabilidad técnica y metodológica de diseñar y evaluar modelos supervisados capaces de estimar calificaciones de

tareas de programación a partir de un conjunto de datos estructurado, sin sustituir el criterio del docente ni asumir que la predicción algorítmica equivale automáticamente a una evaluación pedagógica justa.

Pregunta principal de investigación

¿En qué medida la integración de resultados de pruebas funcionales y métricas de calidad de código permite estimar, mediante modelos de aprendizaje automático supervisado, las calificaciones de tareas de programación con un error predictivo aceptable y utilidad como apoyo al proceso de evaluación docente?

Objetivos Generales y Específicos

Objetivo General

Diseñar, implementar y evaluar un modelo basado en aprendizaje automático para estimar calificaciones de tareas de programación mediante la integración de resultados de pruebas funcionales y métricas de calidad de código.

Objetivos Específicos

- Analizar enfoques existentes de evaluación automática y evaluación asistida de tareas de programación mediante aprendizaje automático y técnicas de análisis de código.
- Caracterizar el conjunto de datos disponible, identificando su fuente, estructura, tamaño, escala de calificación, variables predictoras, variable objetivo y posibles sesgos.
- Definir y preparar las variables de evaluación derivadas de resultados funcionales, métricas de calidad de código y atributos estructurales disponibles en las entregas de programación.
- Implementar modelos supervisados para estimar calificaciones de tareas de programación y compararlos con un modelo base o criterio de referencia.
- Evaluar el desempeño predictivo de los modelos mediante métricas, análisis de errores e interpretación de variables relevantes.
- Establecer las condiciones, limitaciones y recomendaciones para el uso del modelo como apoyo al proceso de calificación docente.

Justificación del proyecto

Dentro de la formación en carreras afines a la tecnología, la evaluación de tareas de programación constituye un proceso fundamental para valorar competencias asociadas al pensamiento lógico, la resolución de problemas, el diseño algorítmico y la aplicación de buenas

prácticas de desarrollo de software (Messer et al., 2024). Por esta razón, la revisión de estas actividades no solo implica verificar si el programa produce una salida correcta, sino también valorar una gran cantidad de criterios que convierten en muchos casos a la calificación manual en una actividad extensa, demandante e ineficiente; especialmente en cursos con alta cantidad de estudiantes o entregas frecuentes. De hecho, estudios recientes señalan que revisar una tarea de programación en cursos introductorios universitarios puede tomar alrededor de treinta minutos por estudiante, por lo que la evaluación manual de tareas de programación puede representar una carga considerable para los docentes (Shdaifat et al., 2025).

Además, debido al tiempo que demanda al docente la evaluación manual de tareas de programación, la demora en la entrega de retroalimentación también afecta el aprendizaje y el rendimiento académico de los estudiantes. Esto se debe a que, cuando la retroalimentación no se proporciona de manera oportuna, los estudiantes tienen menos posibilidades de identificar y corregir sus errores, reforzar conceptos y mejorar sus siguientes entregas, en este sentido y como consecuencia, continuarán repitiendo las mismas fallas en actividades posteriores, lo que limita su progreso académico y reduce la efectividad del proceso de enseñanza-aprendizaje. De hecho, la literatura sobre herramientas de evaluación automática en programación destaca que la retroalimentación rápida permite que los estudiantes reconozcan errores y realicen nuevos intentos de mejora, lo cual favorece su proceso de aprendizaje (Messer et al., 2023). Por estas razones, resulta necesario explorar mecanismos que apoyen al docente en la revisión de tareas, sin reemplazar su criterio académico, pero sí facilitando una valoración más ágil y consistente.

Por otra parte, en muchos casos, la calificación de tareas de programación se centra en el resultado final del programa o en el cumplimiento general de los requerimientos planteados, sin embargo, este enfoque puede dejar en segundo plano aspectos relevantes del proceso de desarrollo, como la organización del código, la lógica utilizada, la claridad de los nombres, el manejo de errores, la reutilización, la documentación y el cumplimiento de buenas prácticas. En este sentido, un estudiante puede presentar una solución parcialmente correcta o evidenciar una aproximación adecuada al problema, pero recibir una calificación baja si la evaluación se enfoca únicamente en la salida final del programa.

De igual manera, investigaciones recientes señalan que los criterios parcialmente subjetivos, como legibilidad, mantenibilidad y documentación, suelen generar dificultades de consistencia en la evaluación de tareas de programación (Messer, 2022; Messer et al., 2024), lo que puede

producir variaciones en la calificación entre evaluadores o entre entregas similares, afectando la equidad y la objetividad del proceso evaluativo.

Debido a esto, se plantea el diseño de un modelo basado en aprendizaje automático para estimar calificaciones de tareas de programación mediante la integración de resultados de pruebas funcionales y métricas de calidad de código, ya que, este enfoque permitirá representar cada entrega estudiantil a través de variables relevantes asociadas tanto al comportamiento funcional del programa como a características estructurales del código fuente, permitiendo que el modelo pueda considerar no solo si la solución produce el resultado esperado, sino también aspectos relacionados con la organización, complejidad, documentación, reutilización y calidad general del código.

Finalmente, la investigación busca aportar al conocimiento al evaluar la viabilidad de utilizar un modelo de aprendizaje automático para estimar calificaciones de tareas de programación mediante la integración de resultados de pruebas funcionales y métricas de calidad de código. Asimismo, permitirá analizar la relación entre estas variables y las calificaciones de referencia, con el propósito de identificar qué características contribuyen en mayor medida al proceso de estimación. En conclusión, el estudio busca aportar evidencia sobre el uso de enfoques más integrales, sistemáticos y replicables para la evaluación asistida de programación en contextos educativos.

Marco Teórico

Evaluación de tareas de programación

La evaluación de tareas de programación es un proceso académico el cual permite valorar el nivel de aprendizaje alcanzado por los estudiantes en el desarrollo de soluciones, pero esta evaluación no debe ser limitada únicamente a comprobar si el programa realizado por el estudiante entrega la salida esperada, sino también debería considerar la forma en la cual la solución fue construida. Por ello, aspectos como la lógica implementada, la organización del código, la legibilidad, el modularidad, el uso adecuado de estructuras de control y la documentación son elementos relevantes para determinar la calidad de una entrega (Messer et al., 2024).

En el contexto de carreras tecnológicas, las tareas de programación permiten evidenciar las competencias asociadas al pensamiento lógico, la resolución de problemas, la aplicación de buenas prácticas de desarrollo de software, sin embargo, la revisión manual de todos estos

puntos puede resultar compleja cuando existen múltiples entregas, diferentes niveles de avances y criterios técnicos que deben ser analizados de manera detallada. Estos sucesos establecen la necesidad de tener criterios de calificación claros, objetivos y medibles que orienten tanto al docente como al estudiante durante el proceso de evaluación.

Criterios de calificación de programación, normativa y terminología relacionada.

Los criterios de evaluación en programación son los elementos que permiten medir la calidad de una solución desarrollada. Entre los criterios más importantes se encuentran el correcto funcionamiento del programa, la organización del código, la claridad, la modularidad, la reutilización, la simplicidad, el manejo adecuado de errores, la documentación y el cumplimiento de buenas prácticas de programación (ISO/IEC 25010, 2023; Pressman & Maxim, 2020). Estos criterios permiten que la evaluación no se limite únicamente a verificar si el software funciona, sino que también considere la calidad técnica del código y el proceso seguido durante su construcción.

Desde una perspectiva académica, la evaluación debe estar alineada con las directrices de la institución, el plan de estudios de la asignatura, los criterios de valoración y los resultados de aprendizaje establecidos por el docente. En este sentido, cualquier herramienta de apoyo a la evaluación debe respetar los criterios definidos por la institución y funcionar como un complemento al proceso evaluativo, sin reemplazar la decisión final del evaluador.

Machine Learning

El Machine Learning, también conocido como aprendizaje automático, es una disciplina dentro de la inteligencia artificial que posibilita que los sistemas identifiquen patrones a partir de datos y hagan predicciones o estimaciones sin la necesidad de depender únicamente de reglas que se programen manualmente. En lugar de detallar instrucciones específicas para cada escenario posible, los modelos de Machine Learning se valen de datos históricos para detectar las conexiones entre las variables de entrada y un resultado deseado, permitiendo así, que el modelo pueda aplicar lo aprendido a nuevos datos que presenten características similares (Mitchell, 1997; Géron, 2022).

De forma similar, en el ámbito de la investigación en ciencia de datos, el Machine Learning se emplea cuando se dispone de un conjunto de datos que puede ser examinado para descubrir

patrones de utilidad y, dependiendo de la naturaleza del problema, los modelos se pueden agruparse en aprendizaje supervisado, no supervisado o por refuerzo.

Aprendizaje Supervisado

Dentro del aprendizaje supervisado, el problema puede abordarse como una tarea de regresión o de clasificación, dependiendo del tipo de resultado que se desea obtener. Si el objetivo es estimar una calificación numérica para una tarea de programación, el enfoque corresponde a un problema de regresión, ya que el modelo aprende a predecir un valor continuo a partir de las características extraídas del código, como resultados de pruebas funcionales, métricas de calidad, estructura del programa y cumplimiento de criterios definidos.

Por otro lado, si las calificaciones se organizan en categorías como desempeño bajo, medio o alto, el problema puede tratarse como una tarea de clasificación. En este caso, el modelo aprende a asignar cada entrega a una categoría específica según los patrones identificados en los datos de entrenamiento.

Para evaluar el desempeño de los modelos de regresión, pueden emplearse métricas como MAE, MSE, RMSE y R^2, las cuales permiten medir el nivel de error entre las calificaciones reales y las calificaciones estimadas por el modelo. En el caso de los modelos de clasificación, pueden utilizarse métricas como exactitud, precisión, recall y F1-score, con el fin de analizar qué tan correctamente el modelo clasifica las entregas en cada nivel de desempeño.

Aprendizaje No Supervisado

Desde el enfoque de aprendizaje no supervisado, el problema puede abordarse mediante técnicas que permitan identificar patrones, agrupaciones o comportamientos similares entre las tareas de programación, sin utilizar una calificación previa como variable objetivo. Este enfoque resulta útil cuando no se dispone de notas etiquetadas o cuando se desea explorar la estructura interna de los datos antes de construir modelos predictivos.

Una alternativa sería aplicar técnicas de agrupamiento, como clustering, para clasificar automáticamente las entregas en grupos con características similares. Por ejemplo, podrían identificarse grupos de estudiantes cuyas soluciones presentan alta calidad de código, errores frecuentes, bajo cumplimiento de pruebas funcionales o estructuras de programación similares. Estos grupos permitirían comprender mejor los distintos perfiles de desempeño sin necesidad de asignar inicialmente una calificación.

Asimismo, el aprendizaje no supervisado puede utilizarse para detectar casos atípicos, como entregas con comportamientos inusuales, códigos incompletos, soluciones copiadas o resultados que se alejan considerablemente del patrón general. También puede aplicarse para reducir la dimensionalidad de los datos cuando existen muchas variables de evaluación, facilitando así el análisis y la visualización de la información.

En este sentido, el aprendizaje no supervisado no reemplaza directamente la evaluación con calificaciones, pero puede complementar el análisis al permitir descubrir patrones ocultos, segmentar entregas y generar información útil para mejorar la construcción de futuros modelos supervisados. Calificación automatizada de tareas de programación mediante Machine Learning.

La calificación automática de trabajos de programación se vincula al Machine Learning dado que posibilita la transformación del código fuente en información susceptible de análisis para el entrenamiento de modelos predictivos. En este procedimiento, cada entrega efectuada por el estudiante se puede simbolizar a través de variables que se obtienen del análisis del código, la extracción de métricas de calidad y la aplicación de técnicas de minería de texto. Dichas variables pueden abarcar la cantidad de clases, métodos, comentarios, líneas de código, estructuras de control, palabras clave, complejidad y otros componentes ligados a la calidad de la resolución.

Con base en estas particularidades, los modelos de Machine Learning pueden ser utilizados para identificar patrones presentes en entregas de programación previamente evaluadas. De acuerdo con Mitchell (1997), el aprendizaje automático permite que un sistema mejore su desempeño a partir de la experiencia, lo cual, en este contexto, implica aprender de tareas calificadas con anterioridad para estimar la valoración de nuevas entregas. De esta manera, el proceso de evaluación deja de depender únicamente de directrices manuales predefinidas y se orienta hacia una valoración apoyada en datos, en la cual el modelo analiza la relación entre las características del código fuente, los resultados obtenidos en las pruebas funcionales y las puntuaciones asignadas previamente.

Esta perspectiva resulta relevante para la investigación, debido a que diversos estudios sobre evaluación automática de tareas de programación señalan que muchas herramientas se han centrado principalmente en verificar la corrección del programa mediante pruebas unitarias, comparación de salidas o análisis estático del código (Keuning, Jeuring & Heeren, 2018; Messer, Brown, Kölling & Shi, 2024). No obstante, una evaluación más completa debe

considerar también elementos internos del código, tales como legibilidad, mantenibilidad, documentación, estructura, orden y complejidad. En este sentido, métricas como la complejidad ciclomática propuesta por McCabe (1976) permiten analizar la estructura lógica del programa y aportar criterios adicionales para valorar la calidad de una solución.

Por este motivo, el modelo híbrido propuesto busca integrar criterios académicos, resultados de pruebas funcionales, minería de texto, métricas de calidad del código y modelos de aprendizaje automático, con el propósito de estimar calificaciones de manera más exhaustiva. Sin embargo, el uso de Machine Learning no debe entenderse como un sustituto del juicio del docente, sino como una herramienta de apoyo que contribuye a optimizar la eficacia, imparcialidad y uniformidad del proceso de evaluación.

Metodología de la Investigación

La investigación se organizará según la metodología CRISP-DM, debido a que permite estructurar proyectos de aprendizaje automático desde la comprensión del problema hasta la evaluación de los resultados. En la fase de comprensión del problema se delimitará el uso del modelo como una herramienta de apoyo a la evaluación docente, sin reemplazar el criterio del profesor. En la fase de comprensión de los datos se analizará la fuente, tamaño, estructura, escala de calificación, variables disponibles, distribución de notas, valores faltantes, duplicados y posibles sesgos del dataset.

Posteriormente, en la fase de preparación de datos se aplicarán procedimientos de limpieza, transformación, codificación, normalización o estandarización cuando corresponda, además de selección de características y control de posibles fuentes de fuga de información. En la fase de modelado se implementarán modelos supervisados y se compararán con un modelo base. En la fase de evaluación se analizará el desempeño mediante distintas métricas priorizando aquellas que permitan una mejor interpretabilidad y comprender cual es el error de la predicción respecto a la nota real.

Finalmente, los resultados se presentarán mediante tablas, gráficos comparativos e interpretación de variables relevantes, con el propósito de valorar la viabilidad técnica y metodológica del modelo como apoyo al proceso de evaluación docente.

Diseño de la Investigación

La investigación se desarrollará bajo un enfoque cuantitativo, con finalidad aplicada, diseño no experimental y alcance predictivo-comparativo. El enfoque será cuantitativo porque se utilizarán variables derivadas de las entregas de programación y métricas estadísticas para evaluar el desempeño de los modelos de aprendizaje automático. La finalidad será aplicada porque el estudio busca proponer una solución orientada a apoyar una necesidad práctica del contexto educativo: la estimación de calificaciones de tareas de programación.

Por otra parte, el diseño será no experimental, debido a que no se manipularán variables independientes, no se asignarán tratamientos a los estudiantes ni se modificarán las condiciones del proceso de enseñanza-aprendizaje.

Finalmente, el alcance será predictivo-comparativo, porque se entrenarán y contrastarán varios modelos supervisados de aprendizaje automático con el propósito de estimar calificaciones. La comparación entre modelos permitirá determinar cuál presenta mejor desempeño predictivo frente a un modelo base.

Cronograma de Actividades

Actividades	Junio Sem 3	Junio Sem 4	Julio Sem 1	Julio Sem 2	Julio Sem 3	Julio Sem 4	Agosto Sem 1	Agosto Sem 2
Comprensión del negocio	x							
Revisión sistemática de literatura	x							
Definición de criterios		x						
Exploración de datos		x	x					
Preparación y limpieza de datos			x	x	x			
Modelado					x	x		
Evaluación, validación y comparación de modelos seleccionados						x	x	
Redacción de los resultados encontrados								x

Alcance, delimitaciones y limitaciones previstas

- El estudio se limitará al dataset proporcionado por el tutor académico; por tanto, los resultados no podrán generalizarse automáticamente a todos los cursos, lenguajes de programación o instituciones.

- La investigación abordará la estimación de calificaciones como problema de regresión supervisada, no como reemplazo de la evaluación docente.
- La calidad de las predicciones dependerá de la consistencia de las calificaciones de referencia y de la pertinencia de las variables disponibles.
- Si el dataset contiene pocas observaciones, muchos atributos o varias entregas similares, se deberá controlar el riesgo de sobreajuste y fuga de información.
- No se realizará inferencia causal sobre el aprendizaje estudiantil; los resultados solo permitirán evaluar desempeño predictivo y utilidad potencial del modelo.

Recursos requeridos

Categoría	Recurso	Uso previsto
Humanos	Tesista	Preparación de datos, implementación de modelos, análisis de resultados y redacción del documento.
Humanos	Tutor académico	Orientación metodológica, validación del alcance y revisión de decisiones técnicas.
Técnicos	Python	Procesamiento de datos, modelado y evaluación.
Técnicos	Scikit-learn	Implementación de modelos supervisados y métricas de regresión.
Técnicos	Pandas y NumPy	Manipulación, limpieza y análisis de datos.
Técnicos	Jupyter Notebook	Experimentación, documentación del flujo de trabajo y reproducibilidad.
Técnicos	Herramientas de análisis estático de código	Extracción de métricas de calidad, si no están incluidas en el dataset.
Técnicos	Git/GitHub	Control de versiones del código y documentación del proyecto.
Bibliográficos	Bases académicas y gestor bibliográfico	Revisión de literatura y normalización de referencias en APA 7.