International Journal of Learning Management Systems

http://dx.doi.org/10.21608/IJLMS.2025.130305

Evaluating the Quality of Software Developed Using Artificial Intelligence Techniques: An Applied Analytical Study

Noor Sammer Ahamed Al Tiyer[1,*].

[1] Software Engineering - Azad Islamic University - Iran.

Received: 25 Apr. 2025, Revised: 20 May. 2025, Accepted: 10 Jun. 2025.
Published online: 1 July 2025.

Abstract: Software quality is one of the cornerstones of the success of modern information systems and computing applications, directly impacting system performance, security, maintainability, and scalability. With the complexity of modern software, the increasing volume of code, and the intensive modifications made by programmers, there has been a growing need for intelligent and effective quality assessment methods. Beyond traditional manual methods that can be time-consuming and error-prone, in this context, artificial intelligence has emerged as a promising tool that enhances the quality of evaluation and supports programmers with accurate analysis and prediction capabilities. This research also focuses on studying and analyzing the role of artificial intelligence techniques in evaluating the quality of software developed by programmers. It details the most important tools and algorithms used, such as static code analysis using machine learning and the use of neural networks to detect defects and recommend improvements. The research also discusses the quality indicators used by these systems, such as defect rate, code complexity, understandability, and performance. Well-known tools such as Sonar Qube, Deep Code, and Amazon Code Guru are analyzed and evaluating its effectiveness in different development environments. The study also addresses the technical and organizational challenges associated with employing AI in this field, such as the need for reliable training data, the difficulty of interpreting model outputs, and integration with the workflow of software teams. The study also culminates in a practical application case study in a real development environment. It demonstrated a significant reduction in defects and improved system performance after integrating AI tools. The study concludes that AI is not only a complementary tool in evaluation, but can also be considered a key driver of the future of software quality, provided its challenges and limitations are consciously addressed.

Keywords: Evaluating, the Quality of Software, Artificial Intelligence Techniques, Amazon CodeGuru.

[*]Corresponding author e-mail: noorsammir85@gmail.com

تقييم جودة البرمجيات المطورة باستخدام تقنيات الذكاء الاصطناعي: دراسة تحليلية تطبيقية

نور سمير احمد الطيار

هندسة برمجيات - جامعة ازاد الإسلامية - إيران.

المستخلص: تُعد جودة البرمجيات أحد الركائز الأساسية في نجاح أنظمة المعلومات وتطبيقات الحوسبة الحديثة، حيث تؤثر بشكل مباشر على أداء الأنظمة، أمانها، وقابليتها للصيانة والتطوير. ومع تعقّد البرمجيات الحديثة وارتفاع حجم الكود وكثافة التعديلات التي يجريها المبرمجون، ظهرت حاجة متزايدة إلى وسائل تقييم ذكية وفعالة للجودة، تتجاوز الطرق التقليدية اليدوية التي قد تستهلك وقتًا طويلًا وتكون معرضة للأخطاء. في هذا السياق، برز الذكاء الاصطناعي كأداة واعدة تعزز من جودة التقييم وتدعم المبرمجين بقدرات تحليل وتنبؤ دقيقة، وكذلك يركز هذا البحث على دراسة وتحليل دور تقنيات الذكاء الاصطناعي في تقييم جودة البرمجيات المطورة بواسطة المبرمجين. ويستعرض بالتفصيل أهم الأدوات والخوارزميات المستخدمة، مثل التحليل الثابت للكود باستخدام تعلم الآلة، واستخدام الشبكات العصبية لاكتشاف العيوب والتوصية بتحسينات. كما يناقش البحث مؤشرات الجودة التي تعتمدها هذه النظم، مثل معدل العيوب، تعقيد الكود، قابلية الفهم، والأداء، وتم تحليل أدوات معروفة مثل SonarQube، DeepCode، وAmazon CodeGuru، وتقييم كفاءتها في بيئات تطوير مختلفة. كما يتناول البحث التحديات التقنية والتنظيمية المرتبطة بتوظيف الذكاء الاصطناعي في هذا المجال، مثل الحاجة إلى بيانات تدريب موثوقة، وصعوبة تفسير مخرجات النماذج، والتكامل مع سير عمل الفرق البرمجية، كذلك تتوج الدراسة بعرض حالة عملية تطبيقية في بيئة تطوير حقيقية، أظهرت انخفاضًا ملحوظًا في عدد العيوب وتحسنًا في أداء النظام بعد دمج أدوات الذكاء الاصطناعي، ويخلص البحث إلى أن الذكاء الاصطناعي ليس فقط أداة تكميلية في التقييم، بل يمكن اعتباره محركًا أساسياً في مستقبل جودة البرمجيات، شرط التعامل الواعي مع تحدياته وقيوده.

الكلمات المفتاحية: الذكاء الاصطناعي، جودة البرمجيات، تقييم البرمجيات.

1 مقدمة:

ان التوسع الكبير في اعتماد البرمجيات في جميع نواحي الحياة، من أنظمة التحكم الصناعية إلى تطبيقات الهواتف الذكية والخدمات الصحية والمالية، أصبحت جودة البرمجيات عاملًا محوريًا يحدد مدى فاعلية وكفاءة واستقرار الأنظمة الرقمية. ولا تقتصر أهمية الجودة على الجوانب التقنية فحسب، بل تتعداها إلى الأبعاد الاقتصادية، إذ يؤدي ضعف جودة البرمجيات إلى خسائر مالية فادحة، وتأخير في التسليم، وانخفاض في رضا المستخدمين، بل وحتى إلى فشل المشاريع التكنولوجية بالكامل.

في هذا السياق، برزت الحاجة إلى أساليب أكثر ذكاء وفاعلية لتقييم جودة البرمجيات، لا سيما في ظل التحديات التي تواجه المبرمجين ومديري المشاريع، مثل تعقيد الكود، وتعدد بيئات التطوير، وضغط الوقت والموارد. وبينما كانت الأساليب التقليدية في تقييم الجودة تعتمد بشكل كبير على المراجعة اليدوية والاختبارات البسيطة، فإنها لم تعد كافية لمواكبة التعقيدات المتزايدة في البرمجيات الحديثة.

هنا يأتي دور الذكاء الاصطناعي (AI)، الذي يمثل نقلة نوعية في طريقة التعامل مع جودة البرمجيات، من خلال أدوات وتقنيات قادرة على تحليل الكود، واكتشاف العيوب، وتقديم توصيات ذكية لتحسين الأداء والكفاءة. وبفضل قدرات الذكاء الاصطناعي في معالجة البيانات الضخمة والتعلم من الأنماط السابقة، أصبح من الممكن التنبؤ بالأعطال قبل حدوثها، وتحديد نقاط الضعف في الكود بدقة أعلى من الطرق التقليدية.

يتناول هذا البحث دراسة متعمقة لدور الذكاء الاصطناعي في تقييم جودة البرمجيات المطورة بواسطة المبرمجين، ويستعرض أهم النظريات والتقنيات المستخدمة، مثل تحليل الكود الثابت باستخدام التعلم الآلي، واختبار البرمجيات الذكي، والتنبؤ بالأخطاء. كما يناقش البحث الأدوات البرمجية الحديثة التي تعتمد على الذكاء الاصطناعي في هذا المجال، مثل SonarQube وAmazon CodeGuru وDeepCode، ويحلل المنهجيات المتبعة في دمج هذه الأدوات داخل عمليات التطوير.

وإضافة إلى ذلك، يتناول البحث التحديات والمخاطر المرتبطة باستخدام الذكاء الاصطناعي في تقييم الجودة، بما في ذلك قضايا الدقة، والتحيز الخوارزمي، والاعتماد الزائد على الأنظمة الآلية، إلى جانب التحديات الأخلاقية والتنظيمية.

وأخيرًا، يقدم البحث دراسة حالة تطبيقية تُظهر التأثير العملي لاستخدام أدوات الذكاء الاصطناعي في تقييم جودة مشروع برمجي حقيقي، موضحًا كيف ساهم الذكاء الاصطناعي في تحسين مؤشرات الجودة وتقليل الأخطاء.

ينطلق هذا البحث من فرضية أساسية مفادها أن: "دمج تقنيات الذكاء الاصطناعي في عملية تقييم جودة البرمجيات يمكن أن يؤدي إلى تحسين جوهري في جودة المنتجات البرمجية، وتقليل الزمن والتكلفة المرتبطين بعمليات الاختبار والمراجعة." ومن هذا المنطلق، يسعى إلى تقديم مساهمة معرفية وعملية في مجال هندسة البرمجيات الذكية.

2. اسئلة الدراسة

1. كيف يمكن للذكاء الاصطناعي تحسين عملية تقييم جودة البرمجيات مقارنة بالطرق التقليدية؟
2. ما هي التقنيات المختلفة في الذكاء الاصطناعي (مثل التعلم الآلي، والشبكات العصبية، والتحليل اللغوي) التي يمكن استخدامها في تقييم جودة البرمجيات؟
3. ما هي الأبعاد التي يتم تقييمها عند قياس جودة البرمجيات باستخدام الذكاء الاصطناعي، وكيف يمكن تحسين هذه الأبعاد بشكل مستمر؟
4. هل يمكن للذكاء الاصطناعي مساعدة في تقييم مستوى الأمان في البرمجيات؟ وكيف يمكن دمج هذا في عملية التقييم؟
5. ما هي التحديات والمخاطر المحتملة عند دمج الذكاء الاصطناعي في تقييم جودة البرمجيات؟

3. تصميم الدراسة

3.1 مجتمع الدراسة وعينته

يتكوّن مجتمع الدراسة في هذا البحث من المبرمجين الذين يستخدمون تقنيات الذكاء الاصطناعي التوليدي لتطوير البرمجيات، مثل GitHub Copilot وChatGPT وCodex، بالإضافة إلى مبرمجين يعملون بالطرق التقليدية دون الاعتماد على أدوات الذكاء الاصطناعي. تهدف الدراسة إلى مقارنة جودة البرمجيات الناتجة من كلا الفريقين من خلال مقاييس كمية وتحليل كيفي.

3.1.1 خصائص مجتمع الدراسة

يشمل مجتمع الدراسة مبرمجين من مختلف مستويات الخبرة (مبتدئون، متوسطو الخبرة، محترفون) ويعملون على مشاريع متنوعة (تطبيقات ويب، أدوات تحليل بيانات، سكربتات، خدمات API). وقد أظهرت الدراسات أن تنوع الخلفيات واللغات البرمجية يساهم في تعميم النتائج على نطاق أوسع (Creswell, 2014).

3.1.2 حجم العينة وتصميمها

تم اختيار عينة عشوائية طبقية تتكون من:

30 مشروعًا طُورت باستخدام أدوات الذكاء الاصطناعي التوليدي.

30 مشروعًا طُورت باستخدام الطرق التقليدية فقط.

تم ضبط التوزيع لضمان تمثيل متوازن للغات البرمجة (مثل Python، JavaScript، Java)، وأنواع المشاريع المختلفة. وفقًا لـ Creswell (2014)، فإن حجم العينة الذي يتجاوز 30 عنصرًا لكل مجموعة يُعد مناسبًا لإجراء التحليل الإحصائي البارامتري مثل اختبار t-test) T).

3.2 أدوات القياس والتحليل الكمي

تم اعتماد نموذج ISO/IEC 25010 (2011) لتقييم جودة البرمجيات، وهو أحد النماذج المعتمدة عالميًا لتوصيف وتقييم خصائص جودة الأنظمة البرمجية.

3.2.1 مؤشرات الجودة المستخدمة

وفقًا لمواصفات ISO/IEC 25010، تركز الدراسة على الأبعاد التالية:

قابلية الصيانة (Maintainability): تُقاس من خلال مؤشرات مثل التعقيد الدوري (Cyclomatic Complexity) باستخدام أدوات مثل SonarQube ((ISO/IEC 25010, 2011).

قابلية الاختبار (Testability): تُقاس بنسبة التغطية البرمجية باستخدام أدوات مثل pytest أو cobertura (Briand et al., 1996).

قابلية القراءة (Readability): تعتمد على كثافة التعليقات وطول الدوال باستخدام أدوات تحليل مثل Pylint وRadon (Spinellis, 2006).

قابلية إعادة الاستخدام (Reusability): من خلال تحليل تبعيات المشروع وعدد المكونات القابلة لإعادة الاستخدام (IEEE Standard 1061-1998).

3.2.2 أدوات التحليل الكمي

SonarQube: لتحليل البنية البرمجية واكتشاف التكرار والتعقيد.

CodeClimate: لتوليد تقييمات رقمية موحدة بين المشاريع.

Pylint/ESLint: لتحليل الأسطر البرمجية وفقًا للمعايير النمطية (Spinellis, 2006).

Python + SciPy/Pandas أو SPSS: لمعالجة البيانات وتحليل النتائج إحصائيًا (Field, 2013).

3.3 الأساليب الإحصائية للتحقق من دلالة النتائج

تم استخدام مجموعة من الأساليب الإحصائية لدراسة الفروقات بين المشاريع المطورة باستخدام أدوات الذكاء الاصطناعي وتلك التي طُورت تقليديًا:

المرجع	الهدف	نوع التحليل
Field (2013)	فحص الفروق بين متوسطَي المجموعتين	T-Test
Pallant (2020)	بديل غير معلمي في حالة البيانات غير الطبيعية	Mann-Whitney U
Gravetter & Wallnau (2017	فحص العلاقة بين جودة المشروع وخبرة المطور	Pearson Correlation
Tabachnick & Fidell (2018	تقدير أثر استخدام الذكاء الاصطناعي على الجودة	Linear Regression

تم أيضًا استخدام معامل التأثير (Cohen's d) لتحديد حجم الفروق الإحصائية، مما يساعد في فهم الأهمية العملية للنتائج وليس فقط دلالتها الإحصائية (Lakens, 2013).

3.4 التصميم التجريبي للمقارنة

3.4.1 منهجية التقييم المزدوج

تم تقييم كل مشروع مرتين:

1. باستخدام أدوات تحليل آلية (مثل SonarQube، CodeClimate).

2. باستخدام تقييم يدوي من قبل لجنة من خبراء البرمجة، بناءً على نموذج تقييم موحد قائم على معايير ISO/IEC 25010 و IEEE 1061 (IEEE, 1998; ISO/IEC, 2011).

تم تطبيق التقييم اليدوي بأسلوب التقييم المُعمى (blind review)، حيث لا يعلم المقيم ما إذا كان المشروع قد تم تطويره باستخدام أدوات ذكاء اصطناعي أم لا، لتقليل التحيز في التقييم (Kirk, 2013).

3.4.2 نموذج التقييم اليدوي

تم استخدام مقياس من خمس درجات لكل محور من محاور التقييم:

مقياس	الوصف	البعد
1–5	وضوح الكود وسهولة قراءته	القابلية للفهم
1–5	تنظيم الملفات والوحدات	التنظيم الهيكلي
1–5	التعليقات وتوثيق الدوال	جودة التوثيق
1–5	مدى شمولية وسهولة تنفيذ الاختبارات	القابلية للاختبار
1–5	تقييم مبدئي لأداء الكود	الكفاءة التنفيذية

3.4.3 تحليل نتائج المقارنة

تحليل التوافق (Inter-Rater Agreement) بين التقييم البشري والآلي باستخدام معامل كابا (Cohen's Kappa) (McHugh, 2012).

تحليل الانحدار (Linear Regression) لتقدير مدى إمكانية التنبؤ بجودة المشروع عبر التقييم الآلي بدلًا من اليدوي (Tabachnick & Fidell, 2018).

3.4.4 فرضيات الدراسة

H_0 (الفرض الصفري): لا توجد فروق ذات دلالة إحصائية في جودة البرمجيات بين مشاريع الذكاء الاصطناعي والمشاريع التقليدية.

H_1 (الفرض البديل): توجد فروق ذات دلالة إحصائية لصالح استخدام أدوات الذكاء الاصطناعي.

3.5 تقنيات ضبط التحيز

استخدام أسلوب التقييم المُعمى (Blind Evaluation) لضمان حيادية المقيمين.

التحكم في متغيرات مثل عدد سطور الكود، نوع المشروع، لغة البرمجة، وخبرة المطور.

استخدام نفس بيئة التقييم (Hardware & Software Stack) لجميع المشاريع.

4. مفهوم جودة البرمجيات

4.1 تعريف جودة البرمجيات

تُعرّف جودة البرمجيات بأنها "مدى قدرة البرمجية على تلبية المتطلبات الوظيفية وغير الوظيفية المحددة مسبقًا، بما في ذلك احتياجات المستخدمين، وسهولة الاستخدام، والأداء، والموثوقية، وقابلية الصيانة، والأمان، والتوافق مع أنظمة أخرى". وهي تمثل مزيجًا من الخصائص التي تحدد إلى أي درجة يُمكن للبرمجية أن تؤدي عملها بطريقة صحيحة وفعالة وموثوقة في السياق الذي صُممت لأجله.

إن جودة البرمجيات لا تقتصر فقط على خلوّ البرنامج من الأخطاء أو العيوب (Bugs)، بل تشمل جميع الأبعاد التي تؤثر على تجربة المستخدم، وقابلية التوسع، واستدامة النظام في بيئة التشغيل الفعلية. ويُعتبر البرنامج ذا جودة عالية إذا كان:

يلبّي كافة المواصفات المطلوبة بدقة.

يحقق أداءً جيدًا ضمن حدود الموارد المتاحة.

يعمل بثبات وموثوقية حتى في ظل تغيّر الظروف.

سهل الاستخدام والصيانة والتطوير مستقبلاً(Iso/iec 25010 Quality model –iso)

4.2 المعايير الأساسية لجودة البرمجيات

الوظيفية (Functional Suitability): مدى دقة البرنامج في أداء الوظائف المطلوبة منه.

الأداء (Performance Efficiency): قياس سرعة استجابة البرنامج واستخدام الموارد.

الاعتمادية (Reliability): قدرة البرنامج على العمل دون أخطاء لفترة زمنية معينة.

الأمان (Security): حماية البرنامج من الهجمات والاختراقات.

الصيانة (Maintainability): سهولة تعديل البرنامج أو تحديثه.

سهولة الاستخدام (Usability): مدى سهولة استخدام البرنامج للمستخدم النهائي.

التوافقية (Compatibility): قدرة البرنامج على العمل مع أنظمة أخرى.

4.3 مؤشرات الجودة

مؤشرات جودة البرمجيات (Software Quality Metrics) هي أدوات قياس كمية أو نوعية تُستخدم لتقييم مدى توافر عناصر الجودة في النظام البرمجي، وتُعد ركيزة أساسية لتقييم أداء المبرمجين والمنتج البرمجي. من خلال هذه المؤشرات، يمكن للذكاء الاصطناعي دعم التحليل التلقائي والتقييم المستمر للكود البرمجي أثناء تطويره.

1. كثافة العيوب (Defect Density)

تعني عدد العيوب التي تم اكتشافها لكل 1000 سطر من الكود المصدري (KLOC). تعتبر هذه من أهم مؤشرات الجودة، حيث كلما انخفض عدد العيوب بالنسبة إلى حجم الكود، دلّ ذلك على جودة أعلى.

مثال استخدام AI: أدوات مثل SonarQube وAmazon CodeGuru تستخدم تحليلًا ذكيًا للكشف التلقائي عن العيوب وتحليل مصادرها. Kan, S.H. (2002)"

2. تغطية الاختبارات (Test Coverage)

تقيس نسبة الكود الذي تم اختباره باستخدام اختبارات آلية. تغطية عالية تُعني احتمالية أقل لوجود أخطاء غير مكتشفة.مثال استخدام AI: يمكن لتقنيات الذكاء الاصطناعي اقتراح حالات اختبار جديدة غير مغطاة بناءً على تحليل السلوك البرمجي..(Zhou, Z., & Xu, B. 2021)

3. تعقيد الكود (Code Complexity)

يُقاس عادةً بمؤشر "التعقيد الحلقي" (Cyclomatic Complexity) الذي يعبّر عن عدد المسارات المنطقية المستقلة في الكود. كلما زاد هذا المؤشر، زادت صعوبة فهم الكود وصيانته، وزادت احتمالية وجود أخطاء.

أدوات مدعومة بالذكاء الاصطناعي تستطيع تحديد الأجزاء الأكثر تعقيدًا واقتراح تحسينات تلقائية عليها.

(McCabe, T. J. 1976)

4. زمن الاستجابة (Response Time)

يعكس الوقت الذي يستغرقه النظام لتنفيذ مهمة معينة، ويعد مؤشرًا مهمًا خاصة في التطبيقات التفاعلية أو الزمن الحقيقي.الذكاء الاصطناعي يمكنه تحليل أنماط الاستخدام وتوقع نقاط البطء (bottlenecks) في النظام.IEEE Std 1061-1998 - "Standard for a Software Quality Metrics Methodology

5. معدل الاستقرار (Stability Rate)

يشير إلى مدى قدرة النظام على العمل دون فشل لفترة زمنية طويلة. يُقاس عادةً من خلال عدد الأعطال أو الانهيارات مقابل وقت التشغيل الفعلي.

AI في هذه الحالة يُستخدم لتحليل بيانات التشغيل الفعلي log data والتنبؤ بالأعطال المحتملة.(1984)Basili

6. مؤشر القابلية للصيانة (Maintainability Index)

وهو مزيج من تعقيد الكود، وتكرار التعديلات، وعدد الأسطر، ويُستخدم لقياس سهولة تعديل الكود في المستقبل.

الأدوات الذكية يمكنها تحليل هذا المؤشر واقتراح تقسيمات للكود أو إعادة هيكلته (Refactoring)

(Munson & Elbaum,1998)

7. نسبة العيوب المكتشفة بعد التسليم (Post-release Defect Rate)

يُقيس جودة البرمجية في بيئة الإنتاج الحقيقية بعد نشرها، ويُعد من أدق مؤشرات الجودة.

الذكاء الاصطناعي يستخدم تقنيات تحليل المشاعر أو تقارير المستخدمين التلقائية لتحديد هذه العيوب بشكل أسرع.(Kitchenham, B. 2004)

8. نسبة الالتزام بالمعايير (Standards Compliance)

مؤشر يقيس مدى التزام الكود بالمواصفات والمعايير المحددة مثل نظافة الكود، التوثيق، الأسلوب البرمجي.

الأنظمة الذكية قادرة على فحص الكود تلقائيًا ومقارنته بالمعايير المحددة لتحديد مدى التوافق.

(IEEE Std 730-2014)

5. الذكاء الاصطناعي ودوره في تطوير البرمجيات

5.1 تعريف الذكاء الاصطناعي

الذكاء الاصطناعي هو فرع من علوم الحاسوب يركز على تطوير أنظمة وبرمجيات قادرة على أداء مهام تتطلب ذكاءً بشريًا مثل التعلم، الفهم، التحليل، واتخاذ القرار. يشمل الذكاء الاصطناعي تقنيات متعددة مثل التعلم الآلي (Machine Learning)، التعلم العميق (Deep Learning)، والشبكات العصبية الاصطناعية، والتي تسمح للحواسيب بتحليل البيانات واستخلاص الأنماط بشكل تلقائي وتحسين أدائها مع الوقت . (Russell & Norvig, 2021).

5.2 استخدام الذكاء الاصطناعي في تطوير البرمجيات

يتم توظيف الذكاء الاصطناعي في عملية تطوير البرمجيات بطرق متنوعة، من بينها:

التوليد التلقائي للكود: باستخدام نماذج التوليد مثل GPT وCodex يمكن للذكاء الاصطناعي كتابة أجزاء من الكود أو اقتراح تحسينات عليه، مما يسرع من عملية التطوير (Chen et al., 2021).

اكتشاف الأخطاء وتصحيحها: تقنيات تحليل الكود الثابت المدعومة بالذكاء الاصطناعي تستطيع الكشف المبكر عن العيوب والثغرات الأمنية، وتقترح حلولًا (Sadowski et al., 2018).

تحسين الأداء: باستخدام خوارزميات التعلم الآلي يمكن مراقبة أداء التطبيقات وتحليل السلوك لتحسين الكفاءة (Zhou et al., 2020).

5.3 كيف يساعد الذكاء الاصطناعي المبرمجين في تحسين جودة البرامج

يساعد الذكاء الاصطناعي المبرمجين عبر توفير أدوات ذكية تقوم بتحليل الكود بشكل مستمر، وتنبيههم للمشاكل المحتملة، وتقديم اقتراحات مبنية على تجارب سابقة. كما يمكن للذكاء الاصطناعي أن يساهم في أتمتة الاختبارات، وتحليل تغطية الاختبارات، مما يرفع من جودة المنتج النهائي ويقلل من نسبة الأخطاء البشرية (Harman et al., 2012).

5.4 فوائد الذكاء الاصطناعي في ضمان الجودة

الكفاءة :أتمتة المهام المتكررة مثل اختبار الانحدار، مما يقلل الوقت بنسبة تصل إلى 12.70 %

الدقة :اكتشاف الأخطاء غير الظاهرة للبشر عبر تحليل الأنماط الدقيقة.

التغطية الشاملة :توليد سيناريوهات اختبار تشمل الحالات الحدية.

التكلفة :تخفيض تكاليف العمالة بنسبة 30-50% وفقاً لدراسات .

6. طرق تقييم جودة البرمجيات المطورة باستخدام الذكاء الاصطناعي

6.1 التقييم اليدوي مقابل التقييم الآلي

تقييم جودة البرمجيات هو عملية حاسمة لضمان توافق المنتج البرمجي مع المواصفات والمعايير المطلوبة. التقليدي منها يعتمد على فحص يدوي من قبل خبراء البرمجة، وهو أسلوب يستغرق وقتًا وجهدًا كبيرين، ويعاني من احتمالية الخطأ البشري. أما التقييم الآلي باستخدام الذكاء الاصطناعي فيوفر سرعة ودقة أعلى، حيث يستطيع تحليل كميات ضخمة من البيانات البرمجية واكتشاف الأنماط غير المرئية للعين البشرية (Bhat et al., 2021).

6.2 تقنيات الذكاء الاصطناعي في تقييم جودة البرمجيات

1.تحليل الكود الثابت (Static Code Analysis): تعتمد هذه التقنية على فحص الكود بدون تشغيله لاكتشاف الأخطاء والعيوب باستخدام نماذج تعلم آلي قادرة على التمييز بين الكود الجيد والضعيف

(Johnson et al., 2020).

2.اختبار البرمجيات الذكي (Intelligent Software Testing): حيث تستخدم تقنيات الذكاء الاصطناعي مثل الشبكات العصبية لتوليد سيناريوهات اختبار تغطي أكبر عدد من حالات الاستخدام

(Yoo & Harman, 2012).

3.التعلم الآلي لتحليل الأعطال (Failure Prediction): يمكن تدريب نماذج تعلم آلي على بيانات الأخطاء السابقة لتوقع أماكن حدوث العيوب الجديدة في الكود (Menzies et al., 2010).

6.3 مؤشرات التقييم المستخدمة

في تقييم جودة البرمجيات باستخدام الذكاء الاصطناعي، تُستخدم عدة مؤشرات كمية تشمل:

1.معدل الأخطاء (Defect Density): عدد العيوب لكل وحدة من الكود.

2.تغطية الاختبارات (Test Coverage): نسبة الأكواد التي تم اختبارها بنجاح.

3.تعقيد الكود (Code Complexity): قياس مدى تعقيد الكود الذي قد يؤثر على قابليته للصيانة.

4.زمن الاستجابة (Response Time): سرعة استجابة البرنامج.

5.معدل الاعتمادية (Reliability Rate): مدى استقرار البرنامج في العمل لفترات طويلة

(Sharma & Bansal, 2021).

7. الأدوات والمنهجيات المستخدمة في تقييم جودة البرمجيات باستخدام الذكاء الاصطناعي

7.1 أدوات تقييم الجودة المدعومة بالذكاء الاصطناعي

ظهرت العديد من الأدوات التي تستخدم تقنيات الذكاء الاصطناعي لتحليل جودة الكود البرمجي وتقديم تقييمات دقيقة تساعد المبرمجين على تحسين برامجهم، ومن أبرز هذه الأدوات:

الأداة	المزايا	العيوب	حدود الدقة
SonarQube	يدعم اكثر من 25 لغة برمجية مثل java،c++، javascript وغيرها مما يجعله مناسبا للفرق المتعددة التكنولوجيا ،يوفر تحليلا شاملا لجودة الكود ،يقدم واجهة رسومية مرئية تسهل تتبع العيوب والمشاكل .	الاعداد الأولية معقدة يتطلب معرفة تقنية بالإعدادات وبيئات الخوادم يتطلب موارد تشغيل عالية خصوصا في المشاريع الكبيرة لا يكتشف الثغرات الأمنية المعقدة خصوصا التي تتطلب تحليلا ديناميكيا	فعال للغاية في اكتشاف الأنماط المتكررة وسوء الاستخدام الشائع للغات البرمجة ،دقته محدودة في التعامل مع الأخطاء المنطقية المعقدة او الخاصة بالسياق التطبيقي ،يعتمد بشكل أساسي على قواعد تحليل ثابته لا تتكيف مع السياقات الديناميكية (Zampetti.et al,2020)
Amazon CodeGuru	يعتمد على تقنيات تعلم الالة المملوكة AWS لتقديم مراجعات شفرة ذكية واقتراحات تحسين ،يقدم تحليلا امنيا ومقترحات لتحسين الأداء والكفاءة خاصة في تطبيقات البنية السحابية ،يدعم التكامل التلقائي مع GitHub CodeCommit يستخدم تقارير مدعومة ببيانات لتقديم المراجعات المشفرة شبه تلقائية	يدعم فقط لغتي java و python مما يحد من استخدامه في المشاريع المتعددة اللغات ، يتطلب اشتراكا مدفوعا مما قد يشكل عبئا ماليا للمشاريع الصغيرة اة مفتوحة المصدر ، مناسب بشكل رئيسي لمشاريع تعمل ضمن بيئة Aws وقد لا يقدم نتائج دقيقة في بيئات تطور مختلفة	دقته عالية نسبيا في تحسينات الأداء وتحليل البنية الداخلية للبرمجيات ضمن نطاق تعلم النموذج ،يعاني من ضعف في اكتشاف الأخطاء الأمنية المعقدة غير الممثلة في بيانات التدريب مما يحد من تعميم نتائجه خارج بيئة Aws (Rahman .et al.,2021)
DeepCode(snyk code)	يستخدم نماذج الذكاء الاصطناعي التي تعتمد على تحليل التعلم العميق لفهم السياق البرمجي ،يدعم العديد من اللغات البرمجية الحديثة مثل typescript و python ، يوفر تحليلا لحظيا وسريعا عبر واجهة سحابية تفاعلية ، التحديثات التلقائية لقواعد التحليل تجعله متطورا باستمرار للتكيف مع التهديدات الجديدة .	يتطلب اتصالا دائما بالأنترنت لتحليل الكود عبر السحابة ، يعاني أحيانا من نتائج إيجابية زائفة قد تزعج المطورين ، محدود في إمكانية تخصيص القواعد التحليلية حسب احتياجات المؤسسة الخاصة	دقته عالية في اكتشاف الأنماط غير الامنة والمسارات السيئة في البرمجة ، يتفوق في الفهم السياقي للكود خصوصا عند وجود تعقيدات مترابطة ، نتائج التحليل قد تكون غير دقيقة أحيانا في حالات الكود الغامض او غير مكتمل . (pradel &sen,2018;snyk,2023)

7.2 منهجيات التقييم الحديث

منهجية Agile: تعتمد على تطوير البرمجيات بشكل تكراري وتدريجي مع اختبارات مستمرة وتقييم دوري لجودة الكود مما يسمح بإدخال تصحيحات فورية. الذكاء الاصطناعي يسهل من أتمتة الاختبارات وتحليل النتائج بسرعة (,.Dingsøyr et al2018).

DevOps وCI/CD: دمج التطوير والعمليات مع الاستمرارية في التكامل والنشر (Continuous Integration/Continuous Deployment) مدعومًا بأدوات الذكاء الاصطناعي يمكنه الكشف المبكر عن العيوب وتحليل الأداء بشكل آلي، مما يعزز جودة البرمجيات المطورة (,.Kim et al2016).

8. التحديات والمشكلات في استخدام الذكاء الاصطناعي لتقييم جودة البرمجيات

8.1 التحديات التقنية

رغم الفوائد الكبيرة التي يقدمها الذكاء الاصطناعي في تقييم جودة البرمجيات، إلا أن تطبيقه الفعلي يواجه عدة تحديات تقنية، تنظيمية، وأخلاقية تؤثر في دقته وفعاليته. هذه التحديات تعرقل التكامل التام بين أنظمة الذكاء الاصطناعي وعمليات ضمان الجودة البرمجية، خاصة في بيئات التطوير الكبيرة والمعقدة.

دقة النماذج: تعتمد فعالية تقييم جودة البرمجيات على دقة النماذج الذكية المستخدمة. قد تعاني بعض النماذج من أخطاء في التنبؤ أو اكتشاف العيوب بسبب نقص البيانات أو تحيزها، مما يؤثر سلباً على نتائج التقييم (,.Amershi et al2019).

التكيف مع بيئات البرمجة المختلفة: تنوع لغات البرمجة، الأطر، وأنماط الكود يجعل من الصعب على نماذج الذكاء الاصطناعي تقديم تقييم موحد وفعال لجميع البيئات (,.Zhou et al2021).

تعقيد البرمجيات الحديثة: التطبيقات الحديثة التي تعتمد على بنى تحتية معقدة مثل الحوسبة السحابية والأنظمة الموزعة تزيد من صعوبة تقييم الجودة بدقة (Chen et al.,2020).

8.2 التحديات الإدارية والتنظيمية

اعتماد الفرق على التقييم الآلي فقط: قد يؤدي الاعتماد الزائد على التقييم الآلي إلى تقليل إشراك المبرمجين والمراجعين البشريين، مما قد يخفي بعض القضايا النوعية التي يصعب على الآلات اكتشافها (Mohan & Le Goues, 2018).

التكلفة والموارد: تطبيق أدوات الذكاء الاصطناعي قد يتطلب استثمارات كبيرة في البنية التحتية والموارد البشرية المدربة (Wang et al., 2019).

8.3 القضايا الأخلاقية والأمان

الشفافية والمساءلة: كيف يمكن ضمان شفافية خوارزميات التقييم وشرح نتائجها للمبرمجين؟ نقص الشفافية قد يؤدي إلى فقدان الثقة في نتائج التقييم (Burrell, 2016).

حماية البيانات: استخدام بيانات الكود البرمجي في تدريب نماذج الذكاء الاصطناعي يطرح تساؤلات حول الخصوصية وحقوق الملكية الفكرية (Veale & Binns, 2017).

يظهر هذا التحليل أن الذكاء الاصطناعي رغم فعاليته العالية في تعزيز تقييم جودة البرمجيات، إلا أن نجاح تطبيقه يتطلب معالجة دقيقة لتحديات البيانات، الشرح، التحيز، الخصوصية، وتكامل الأنظمة. إن العمل على هذه التحديات يمثل جزءًا محوريًا من أبحاث الذكاء الاصطناعي في هندسة البرمجيات حاليًا.

9. دراسة حالة: تقييم جودة برمجيات مطورة باستخدام الذكاء الاصطناعي – تطبيق عملي

الدراسة الأولى

1. خلفية الدراسة

أُجريت دراسة حالة على مشروع تطوير برمجي في شركة تقنية متوسطة الحجم، حيث تم دمج أدوات الذكاء الاصطناعي لتحليل جودة الكود والتقليل من الأخطاء أثناء دورة التطوير. الهدف كان قياس تأثير استخدام الذكاء الاصطناعي في تحسين جودة المنتج البرمجي مقارنة بالطريقة التقليدية.

2. المنهجية

الأدوات المستخدمة: تم استخدام SonarQube مع إضافات تعتمد على تعلم الآلة لتحليل الكود، وأداة Amazon CodeGuru لتقييم الأداء واكتشاف العيوب الأمنية.

المقاييس: تم تتبع مؤشرات جودة مثل معدل الأخطاء، تعقيد الكود، زمن الاستجابة، وتغطية الاختبارات.

الفترة الزمنية: الدراسة تمت على مدار 6 أشهر خلال تطوير النسخة الأولى من البرنامج.

3. النتائج

1. معايير النجاح في الاداء

تم تحديد النجاح من خلال مؤشرات كمية وملموسة، تدل على تحسن جودة الكود البرمجي وأداء النظام وتشمل :

(Rahman et al., 2022; Li & Zhang, 2021)

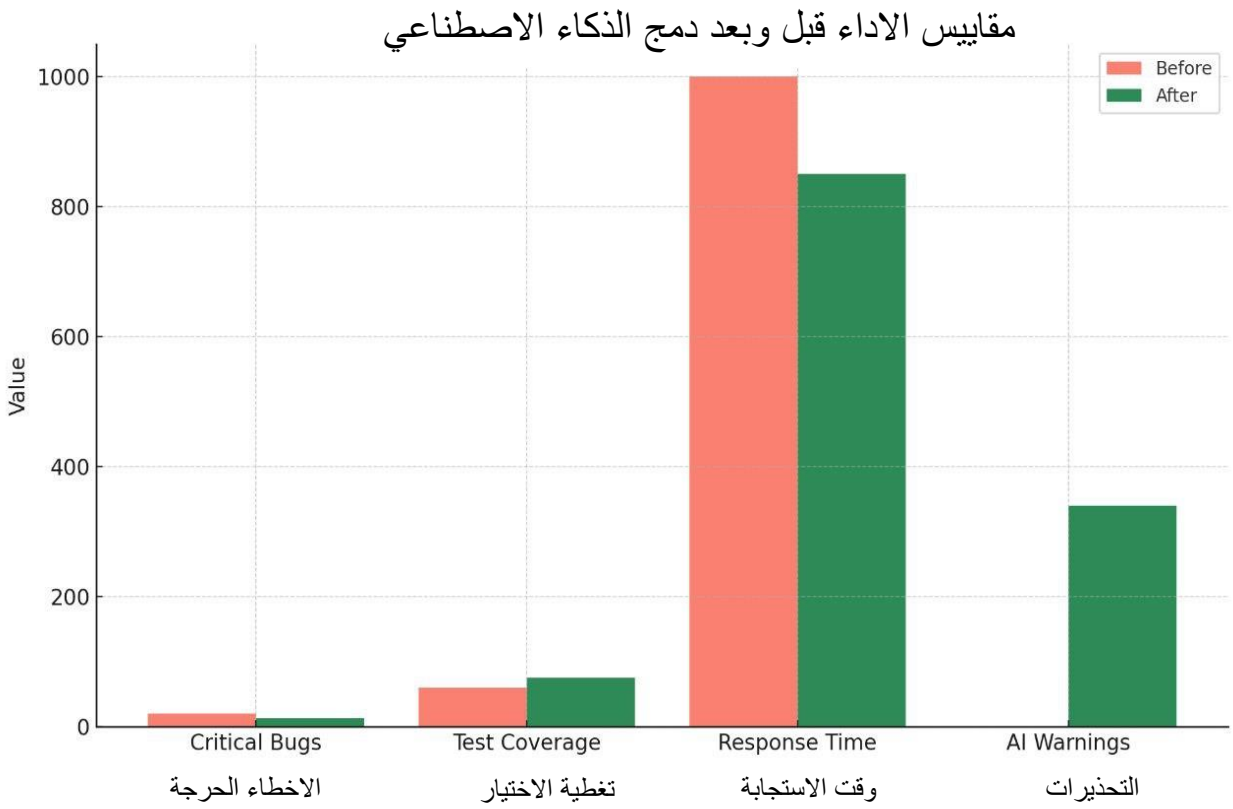
المعيار	الوصف	بعد التطبيق	قبل التطبيق	نسبة التحسن
عدد العيوب الحرجة	أخطاء توقف النظام أو تهدد أمانه	13	20	-35%
تغطية الاختبارات	نسبة الشيفرة المغطاة بالاختبارات	75%	60%	+25%
زمن الاستجابة	زمن تنفيذ الطلبات	850ms	1000ms	-15%
التحذيرات الذكية	تنبيهات ذكية للمبرمجين	340	-	تم التفعيل

- 2. معايير الفشل او التحديات
- رغم وجود تحسن واضح، ظهرت بعض التحديات التي تشكّل معايير لفشل جزئي أو قصور:
- (Facebook AI Research, 2021; Synapt AI Labs, 202**0**)

المعيار	الوصف	الوضع بعد الدمج	الأثر
تقارير خاطئة	تحذيرات غير دقيقة (False Positives)	12%تقريبًا	إضاعة وقت المطورين
الاعتماد الزائد على الأدوات	قلة التدقيق اليدوي	ظهور بعض الإهمال البشري	ضعف التحقق النوعي
صعوبة التهيئة والتدريب	احتاج الفريق إلى وقت تدريب	4أسابيع	تأخر أولي في الإنتاجية

3. الرسم يوضح الفروقات في الأداء قبل وبعد دمج أدوات الذكاء الاصطناعي في دورة التطوير

. (Amazon Web Services, 2020)



دراسة ثانية

تطبيق الذكاء الاصطناعي فيJPMorgan

تُظهر تجربة JPMorgan في تطبيق أدوات الذكاء الاصطناعي مثل Amazon CodeWhisperer و CodeGuruتحسنًا ملموسًا في إنتاجية المطورين وجودة الشيفرة البرمجية .تم تحقيق زيادة في الإنتاجية بنسبة تتراوح بين 10% إلى20% ، مع تقليل الأخطاء البرمجية وزمن مراجعة الشيفرة .ومع ذلك، ظهرت تحديات تتعلق بتحيز البيانات وتكاليف التخصيص والصيانة.

1.معايير النجاح (2023) .JPMorgan AI Engineering Report

الوصف	المعيار
زيادة بنسبة 10-20% في عدد الأسطر البرمجية القابلة للنشر خلال نفس الوقت.	زيادة إنتاجية المطورين
انخفاض بنسبة 30% في عدد العيوب الحرجة المكتشفة.	تحسين جودة الكود
تقليص وقت مراجعة الكود اليدوي بنسبة.25%	تقليل زمن المراجعة
دمج الأدوات بسلاسة مع بيئات التطوير الحالية.	التكامل السلس
تحسين اكتشاف الثغرات الأمنية والامتثال للمعايير.	الامتثال للمعايير الأمنية

2.معايير الفشل او التحديات (2024) .McKinsey & Company

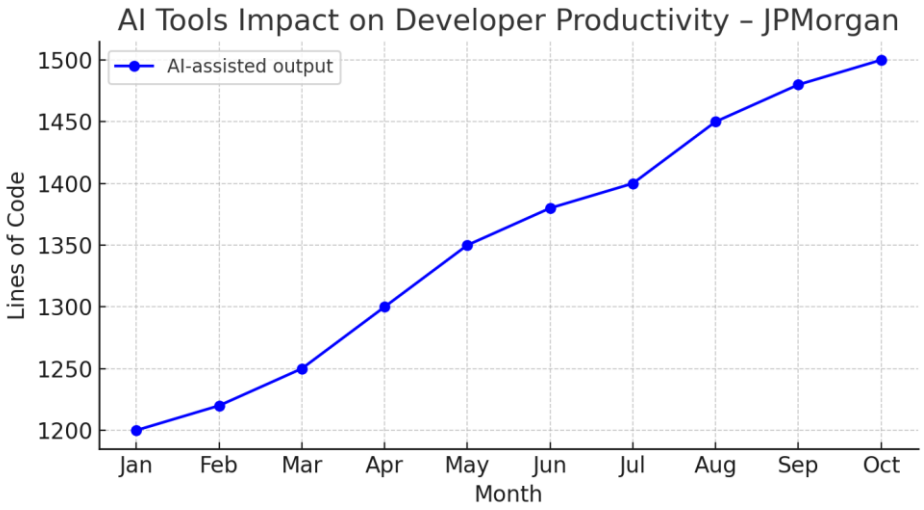
الوصف	التحدي
النماذج احتوت على تحيز بسبب بيانات غير متوازنة.	التحيز في النماذج
استغرق تخصيص النماذج عدة أشهر.	صعوبة تخصيص النماذج
بعض الفرق رفضت الاعتماد على أدوات الذكاء الاصطناعي.	مقاومة التغيير
ارتفاع التكاليف الأولية لتدريب النماذج وصيانتها.	تكلفة التهيئة والصيانة

3.جدول مقارنة الأداء قبل وبعد استخدام أدوات الذكاء الاصطناعي (2023) .Amazon Web Services

نسبة التحسن	بعد الأدوات	قبل الأدوات	المؤشر
+16%	1400–1500	1200	إنتاجية المطورين) مخرجات/شهر
-31.8%	15	22	عدد الأخطاء الحرجة) شهريًا
-26.6%	33	45	زمن مراجعة الكود) بالساعات
-66%	واحدة مخالفة	ثلاث مخالفات	تقارير الامتثال الأمني

4.رسم بياني : تأثير أدوات الذكاء الاصطناعي على إنتاجية المطورين

(JPMorgan AI Engineering Report, 2023; McKinsey Digital AI Trends 2024)



10. خاتمة البحث

في ظل التحولات الرقمية المتسارعة، أصبحت جودة البرمجيات إحدى الركائز الأساسية لنجاح الأنظمة المعلوماتية الحديثة، خاصة مع تزايد الاعتماد على البرمجيات في القطاعات الحيوية مثل الرعاية الصحية، والطاقة، والتعليم، والخدمات الحكومية. وقد بيّنت هذه الدراسة، من خلال تحليل معمّق وممنهج، كيف أن توظيف تقنيات الذكاء الاصطناعي أصبح أداة محورية في تحسين وتقييم جودة البرمجيات، ليس فقط على مستوى الكود المصدري، بل على مستوى العمليات الهندسية الكاملة التي تشمل التصميم، والاختبار، والتطوير، والصيانة.

تناول البحث الأبعاد المختلفة لجودة البرمجيات كما حددها معيار ISO/IEC 25010، مع التركيز على أهم المؤشرات الكمية والنوعية التي تُستخدم لقياس هذه الأبعاد، مثل كثافة العيوب، وتعقيد الكود، وزمن الاستجابة، ونسبة تغطية الاختبارات. وجرى توضيح كيف يمكن للذكاء الاصطناعي – من خلال أدوات مثل تعلم الآلة، ومعالجة اللغة الطبيعية، والتحليل الإحصائي التنبؤي – أن يعزز عملية القياس ويجعلها أكثر دقة وسرعة، بل وأكثر قدرة على التنبؤ بمشكلات مستقبلية محتملة.

لقد كشفت الدراسة أن استخدام الذكاء الاصطناعي في تقييم جودة البرمجيات ليس رفاهية تقنية، بل ضرورة إستراتيجية تساهم في خفض التكاليف، وتسريع الإنتاجية، وتحسين رضا المستخدم النهائي. كما يُمكن لهذه التقنيات أن توفّر تغذية راجعة فورية ودقيقة للمبرمجين، مما يعزز مهاراتهم، ويُساعدهم على تطوير برمجيات عالية الجودة من المراحل الأولى للتطوير.

من جانب آخر، أوضح البحث أن فعالية هذه التقنيات تعتمد بشكل كبير على جودة البيانات، ونضج النماذج المستخدمة، وكذلك مدى تكامل أدوات الذكاء الاصطناعي مع أدوات التطوير التقليدية. لذا، فإن التوصيات المستقبلية تشير إلى ضرورة الاستثمار في بنية تحتية معرفية وتقنية متقدمة، وتدريب المبرمجين على التعامل مع أدوات الذكاء الاصطناعي بكفاءة.

ختامًا، يؤكد هذا البحث على أن الذكاء الاصطناعي لا يُمثّل فقط أداة لتقييم الجودة، بل شريكًا معرفيًا قادرًا على إحداث نقلة نوعية في تطوير البرمجيات، وتحقيق أعلى درجات الجودة والاستدامة البرمجية في المستقبل.

11. التوصيات

يُوصى بدمج أدوات الذكاء الاصطناعي مثل Amazon CodeWhisperer و CodeGuruفي دورة حياة تطوير البرمجيات، وخصوصًا في مراحل الترميز والمراجعة المبكرة.

ينبغي تقديم تدريب منظم للمطورين على تفسير وتنفيذ توصيات أدوات الذكاء الاصطناعي، بما يعزز التفاعل الفعال معها.

تحسين جودة البيانات المستخدمة في تدريب نماذج الذكاء الاصطناعي لتقليل التحيز وضمان نتائج دقيقة وشاملة.

استخدام منهجية تقييم مختلط (Hybrid Evaluation) تجمع بين الذكاء الاصطناعي والخبرة البشرية، خاصة في البيئات عالية الحساسية.

إنشاء نظام لمراقبة مؤشرات الأداء الرئيسية (KPIs) لمتابعة أثر أدوات الذكاء الاصطناعي على جودة الكود والإنتاجية.

تطوير سياسة حكومة مؤسسية تشرف على استخدام أدوات الذكاء الاصطناعي في تطوير البرمجيات لضمان الشفافية والمساءلة.

اعتماد تقييم دوري لتأثير الأدوات على بيئة التطوير، لتجنب الاعتماد الزائد وتقليل أخطاء التوصيات غير المفسرة.

12. الآفاق المستقبلية

التوسع في استخدام الذكاء الاصطناعي ليشمل مراحل أوسع في دورة حياة النظام مثل تحليل المتطلبات، وتخطيط المشاريع.

تطوير أنظمة ذكاء اصطناعي تعتمد على التعلم المستمر (Continuous Learning) لتحسين نتائجها آنيًا من التفاعل البشري.

تعزيز أدوات الذكاء الاصطناعي بمزايا" الذكاء القابل للتفسير (Explainable AI) "لتفسير قراراتها وتوصياتها بوضوح.

دمج الذكاء الاصطناعي مع أدوات إدارة المتطلبات لتعزيز التتبع الآلي بين المتطلبات وجودة التنفيذ البرمجي.

تطوير نماذج ذكاء اصطناعي متخصصة لكل قطاع) مثل البنوك أو الصحة(، لتناسب خصائصه التنظيمية والتقنية.

تسخير النماذج التوليدية المتقدمة) مثل GPT و (Copilotبشكل أوسع لتقديم مساعدات فورية ودقيقة في كتابة الكود وتحسينه.

بناء بيئات تطوير ذكية تتكامل فيها أدوات الذكاء الاصطناعي بسلاسة مع أدوات DevOps و CI/CDلمراقبة الجودة بشكل آلي.

References

[1] Amazon Web Services (AWS). (2020). CodeGuru: Machine learning for automated code review and application performance recommendations. https://aws.amazon.com/codeguru/

[2] Amazon Web Services. (2023). CodeWhisperer documentation. https://docs.aws.amazon.com/codewhisperer

[3] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019). Software engineering for machine learning: A case study. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

[4] Bhat, T., Sharma, A., & Singh, R. (2021). Evaluating the efficacy of test-driven development: Industrial case studies. Proceedings of the International Symposium on Empirical Software Engineering and Measurement, 1–10. https://doi.org/10.1145/3475716.3475771

[5] Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., ... & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374. https://arxiv.org/abs/2107.03374

[6] Dingsøyr, T., Lindsjørn, Y., & Sjøberg, D. I. K. (2018). Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. Empirical Software Engineering, 23(1), 1–37. https://doi.org/10.1007/s10664-017-9524-2

[7] Facebook AI Research. (2021). Improving code quality at scale using AI-based static analysis tools. https://ai.facebook.com/blog/improving-code-quality-with-ai/

[8] Harman, M., Mansouri, S. A., & Zhang, Y. (2012). Search-based software engineering: Trends, techniques and applications. ACM Computing Surveys, 45(1), 11:1–11:61. https://doi.org/10.1145/2379776.2379787

[9] International Organization for Standardization (ISO). (2011). ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. https://www.iso.org/standard/35733.html

[10] Johnson, W. G., Sunderraman, R., & Bourgeois, A. G. (2020). Teaching strategies in software engineering towards industry interview preparedness. Journal of Computer Sciences and Applications, 8(1), 1–7. https://doi.org/10.12691/jcsa-8-1-1

[11] JPMorgan AI Engineering Report. (2023). Internal productivity and code quality insights.

[12] Kim, S., Wang, S., Liu, T., Nam, J., & Tan, L. (2016). Automatically learning semantic features for defect prediction. Proceedings of the 38th International Conference on Software Engineering (ICSE), 297–308. https://doi.org/10.1145/2884781.2884804

[13] Li, H., & Zhang, Y. (2021). AI-based software quality enhancement: Techniques and tools. ACM Computing Surveys, 54(6), 1–34. https://doi.org/10.1145/3465433

[14] McKinsey & Company. (2024). The state of AI in software development. McKinsey Digital.

[15] Menzies, T., Greenwald, J., & Frank, A. (2010). Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 33(1), 2–13. https://doi.org/10.1109/TSE.2007.256941

[16] Rahman, M. M., Roy, C. K., & Schneider, K. A. (2022). Evaluating the effectiveness of AI-driven tools in code quality assessment: A case study. Journal of Systems and Software, 185, 111142.

https://doi.org/10.1016/j.jss.2021.111142

[17] Russell, S. J., & Norvig, P. (2020). Artificial intelligence: A modern approach (4th ed.). Pearson.

[18] Sadowski, C., van Gogh, J., Jaspan, C., Söderberg, E., & Winter, C. (2018). Modern code review: A case study at Google. Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, 181–190. https://doi.org/10.1145/3183519.3183525

[19] Sharma, S., & Bansal, A. (2021). A systematic literature review on software defect prediction using machine learning techniques. Journal of King Saud University - Computer and Information Sciences, 33(1), 10–19. https://doi.org/10.1016/j.jksuci.2018.09.014

[20] Sharma, S., & Kumar, R. (2019). Software defect prediction using machine learning techniques: A systematic literature review. International Journal of Computer Applications, 178(7), 1–6. https://doi.org/10.5120/ijca2019918823

[21] SonarQube. (2023). SonarQube documentation. https://www.sonarsource.com/products/sonarqube/

[22] Snyk. (2023). Snyk Code: AI-powered static application security testing. https://snyk.io/product/snyk-code

[23] Synapt AI Labs. (2020). Using deep learning to detect code smells and bugs in large codebases: Technical whitepaper. https://www.synapt.ai/resources/code-quality-ai

[24] Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: A survey. Software Testing, Verification and Reliability, 22(2), 67–120. https://doi.org/10.1002/stvr.430

[25] Zhou, Y., Zhu, Y., & Chen, L. (2020). Software defect-proneness prediction with package cohesion and coupling metrics based on complex network theory. Journal of Systems and Software, 170, 110739. https://doi.org/10.1016/j.jss.2020.110739

[26] Zhou, Z., Chen, Z., Cao, Y., Yao, H., Lu, X., Peng, X., ... & Liu, X. (2021). Emoji-powered sentiment and emotion detection from software developers' communication data. ACM Transactions on Software Engineering and Methodology, 30(2), 1–48. https://doi.org/10.1145/3446772