Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40350416
Article received 01.08.2025
Article accepted 26.06.2025
UDC 004.[05+8]
Correspondence author: O. I. Torskyi oresttorskyi@gmail.com

***О. І. Торський, Ю. І. Грицюк***
*Національний університет "Львівська політехніка", м. Львів, Україна*

# ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ МОДЕЛЕЙ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Наведено особливості застосування методів машинного навчання ML (англ. *Machine Learning*) моделей для автоматизованого тестування програмного забезпечення (ПЗ) у контексті сучасних практик CI/CD (англ. *Continuous Integration* / *Continuous Delivery*). Виявлено, що підходи, засновані на глибокому навчанні, навчанні з підкріпленням та використанні історичних логів демонструють найбільшу ефективність їх застосування. З'ясовано, що поєднання ML-моделей з метаевристичними алгоритмами здатне покращити процес автоматичного генерування тестів й істотно зменшити кількість нерелевантних сценаріїв тестування ПЗ. Охарактеризовано закономірності використання класифікації тестових сценаріїв, кластеризації логів і скорочення обсягу даних у процесах тестування ПЗ. Оцінено вплив обмежень, таких як недостатній обсяг навчальних даних, складність інтеграції у реальні процеси CI/CD та обмежена масштабованість моделей на якість застосування ML у тестуванні ПЗ. Здійснено змістовний аналіз та узагальнення наукових праць у період з 2012 по 2025 рр. для окреслення перспективних етапів подальшого дослідження. Визначено доцільність подальшого вивчення інструментів машинного навчання моделей66 для динамічного оновлення програмних продуктів і значного зменшення витрат на тестування ПЗ. Результати дослідження підтвердили, що застосування адаптивних моделей онлайн-навчання дає змогу підвищити гнучкість тестових систем у динамічних середовищах розроблення ПЗ. Особливу увагу приділено підходам до пріоритизації тестів на підставі історичних даних та CI-логів, оскільки це сприяє швидкому виявленню помилок у нових версіях ПЗ. У роботі зроблено акцент на практичній значущості гібридних підходів, які поєднують можливості машинного навчання моделі та алгоритмів евристичного пошуку адекватних тестових сценаріїв. Виокремлено основні труднощі, з якими стикаються дослідники, зокрема – потреба у якісній підготовці даних та ефективному обробленні великих обсягів телеметрії. Зазначено, що подальший розвиток у цій галузі вимагає тісної взаємодії між тестувальниками, аналітиками даних та розробниками тестових інтелектуальних систем.

**Ключові слова:** глибоке навчання моделі; навчання з підкріпленням; пріоритизація сценаріїв тестування ПЗ; CI/CD-процеси; гібридні підходи.

## Вступ / Introduction

У сучасному інформаційному суспільстві, де розроблення програмного забезпечення (ПЗ) відбувається швидкими темпами, та за умови постійних змін інформаційного середовища, виникає потреба у пошуку ефективних рішень для забезпечення належної якості відповідних програмних продуктів. Відомо, що автоматизоване тестування ПЗ вже давно стало невід'ємною частиною процесу його розроблення. Однак, очевидно, традиційні підходи до автоматизованого тестування ПЗ вимагають значних ресурсів на розроблення тестів для їхньої підтримки та мають обмежену здатність адаптуватися до змін інтерфейсу тестувальника. Тому виникає потреба у використанні більш гнучких та інтелектуальних підходів, здатних на автоматичний аналіз тестових сценаріїв та їхню адаптацію до різноманітних змін. Одним із таких перспективних напрямів є застосування методів машинного навчання у сфері автоматизованого тестування ПЗ, які забезпечують нові можливості для підвищення ефективності процесу його тестування.

Машинне навчання моделі автоматизованого тестування ПЗ відкриває нові можливості для підвищення ефективності процесу його тестування. Застосування інтелектуальних алгоритмів дає змогу автоматично аналізувати код, дані користувачів, історію помилок та логів. У такий спосіб процес розроблення, вибору та виконання тестів стає досконалішим. Використовуючи методи машинного навчання моделей автоматизованого тестування ПЗ, знижується навантаження на команди QA (англ. *Quality Assurance* – забезпечення якості), забезпечується вища точність виявлення дефектів і швидша реакція на певні зміни інформаційного середовища. Отже, дослідження можливості інтеграції методів машинного навчання моделей автоматизованого тестування ПЗ є актуальним і перспективним напрямом у сфері забезпечення його якості.

Хоча методи машинного навчання моделей вже давно застосовують в окремих етапах автоматизованого тестування ПЗ, проте їхній повний потенціал ще не ре-

**Інформація про авторів:**
**Торський Орест Ігорович**, магістр кафедри інформаційних систем та мереж. **Email:** orestorskyi@gmail.com; https://orcid.org/0009-0002-1896-1021
**Грицюк Юрій Іванович**, д-р техн. наук, професор кафедри програмного забезпечення. **Email:** yurii.i.hrytsiuk@lpnu.ua; https://orcid.org/0000-0001-8183-3466
**Цитування за ДСТУ:** Торський О. І., Грицюк Ю. І. Застосування машинного навчання моделей для підвищення ефективності автоматизованого тестування програмного забезпечення. Науковий вісник НЛТУ України. 2025, т. 35, № 4. С. 142–149.
**Citation APA:** Torskyi, O. I., & Hrytsiuk, Yu. I. (2025). Application of machine learning to enhance the efficiency of automated software testing. *Scientific Bulletin of UNFU*, 35(4), 142–149. https://doi.org/10.36930/40350416

алізований. Саме тому дослідження цього підходу можна вважати безперечно актуальним і перспективним.

*Об'єкт дослідження* – автоматизоване тестування програмного забезпечення.

*Предмет дослідження* – методи і засоби автоматизованого тестування програмного забезпечення, які дають змогу зменшити кількість ручних перевірок його компонент, а також забезпечують стабільнішу роботу за реальних умов його експлуатації.

*Мета роботи* – проаналізувати можливості застосування машинного навчання моделей для підвищення ефективності автоматизованого тестування програмного забезпечення, що дасть змогу отримати максимум інформації про його якість роботи в межах виділених часових ресурсів проєкту, зменшити кількість як дрібних, так і більших дефектів, а також забезпечить стабільнішу роботу продукту проєкту.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- проаналізувати наявні підходи та алгоритми машинного навчання моделей, які використовують для автоматизованого тестування ПЗ, що дасть змогу визначити їхню ефективність та сферу практичного застосування у реальних проєктах;
- виявити ключові особливості впровадження машинного навчання моделі для організації процесу автоматизованого тестування ПЗ, що забезпечить підвищення якості та скорочення часу на випуск оновлень;
- оцінити переваги та можливі обмеження застосування машинного навчання моделей у межах CI/CD-процесів, що допоможе обґрунтувати доцільність його впровадження у конкретних корпоративних середовищах;
- запропонувати узагальнену модель інтеграції методів машинного навчання у тестувальні середовища ПЗ, яка уможливить гнучке масштабування процедури тестування, адаптацію до динамічних змін у кодовій базі та зниження витрат на супровід тестів.

Виконання поставлених завдань сприятиме розробленню ефективніших та гнучкіших систем автоматизованого тестування ПЗ. Водночас це допоможе підвищити якість ПЗ та впорядкувати роботу розробників.

**Аналіз останніх досліджень та публікацій**. Наразі у наукових дослідженнях дедалі більше уваги приділяють застосуванню методів машинного навчання моделей для організації процесу автоматизованого тестування ПЗ. Це зумовлено можливістю машинного навчання вирішувати низку проблем, пов'язаних із ручною підтримкою тестів та обробленням великих обсягів даних.

У роботі [22] автори здійснюють детальний аналіз сучасних методів машинного навчання моделей автоматизованого тестування ПЗ. Особливо детально було зроблено акцент на класифікації та кластеризації цих підходів. Робота також вказує на перспективи інтеграції цих методів у CI/CD-процеси для покращення точності та адаптивності процесу тестування ПЗ. Автори також аналізують PCA (англ. *Principal Component Analysis*) алгоритм для зниження обсягу даних, а також SVM (англ. *Support Vector Machine*), KNN (англ. *k-Nearest Neighbors*) і ACO (англ. *Ant Colony Optimization*) для класифікації даних та автоматизації процесу тестування ПЗ. Чималу увагу приділяють інтеграції цих методів у CI/CD-процеси для того, щоб зробити процес тестування ПЗ більш адаптивним, швидким і точним. Окрім цього, у роботі розглянуто перспективи застосування метаевристичних алгоритмів прийняття ефективних рішень під час генерування тестів і виявлення помилок.

У дослідженні [10] наведено підхід, який поєднує методи машинного навчання моделей з аналізом потоків даних під час безперервної інтеграції ПЗ. Можна вважати, що таке поєднання є досить доцільним і ефективним, адже виникає можливість виявлення аномалій у процесі тестування ПЗ у реальному часі. Автори наголошують, що використання алгоритмів онлайн-навчання дає змогу адаптувати тестові моделі до швидких змін у програмному коді, мінімізуючи затримки його аналізу та підвищуючи якість процесу тестування ПЗ. Проведені експерименти демонструють, що така інтеграція знижує кількість хибнопозитивних результатів і покращує виявлення критичних дефектів. Це свідчить про те, що машинне навчання моделі може ефективно підтримувати автоматизацію процесу тестування ПЗ у динамічних і високонавантажених системах. Зокрема, розглядають статистичні, класичні та нейронні мережі для виявлення аномалій у багатовимірних даних. Дослідження також показує, що такі методи дають змогу ефективно виявляти непередбачувані помилки та покращувати якість процесу тестування ПЗ в CI/CD середовищах.

Актуальність проблеми застосування машинного навчання моделей автоматизованого тестування ПЗ підтверджено також іншими науковими джерелами. Так, у праці [28] наведено фреймворк DeepRNG (англ. *Deep Reinforcement Learning*), який використовує методи глибокого навчання з підкріпленням для автоматизованого генерування тестових сценаріїв. Застосування такого підходу зменшує кількість пропущених помилок і значно покращує ефективність процесу тестування великих програмних систем.

У систематичному огляді праць [2] проаналізовано понад 40 досліджень зі застосування RL (англ. *Reinforcement Learning*) у сфері тестування ПЗ. У роботі охоплено чимало напрямів: функціональне тестування ПЗ, тестування ігор, регресійне тестування ПЗ та автоматичне оновлення тестів. Попри висвітлену перспективність цих методів, автори також акцентують увагу на різноманітних викликах, серед яких вони виділяють обмежений доступ до навчальних даних, потреба у гнучкій інтеграції з CI/CD, а також недостатня кількість досліджень, що використовують складні архітектури RL за промислових умов.

Чимало інших дослідників [9, 19, 25] також вказують на доцільність використання машинного навчання моделі для побудови нових тестів. Зокрема, у дослідженні [25] розглянуто особливості застосування алгоритмів глибокого навчання з підкріпленням для визначення найкритичніших тест-кейсів. Такий підхід дає змогу скоротити тривалість процесу тестування ПЗ, залишаючи якість його перевірки. Так, у проєктах із частими релізами такий спосіб буде надзвичайно корисним, адже потреба оновлювання тестової моделі виникає доволі часто.

У роботі [9] автори проаналізували сучасні тенденції інтеграції машинного навчання моделі у процес автоматизованого генерування тестів. Як наслідок, дослідники виявили, що найчастіше застосовують методи супервізованого навчання (зокрема, нейронні мережі), підкріплення (на підставі Q-learning) та напів- або несупервізовані підходи. Автори окреслили головні виклики: обмежений доступ до якісних навчальних даних, масштабованість і складність повторного тренування та

оцінювання моделей. Також було наголошено на важливості розроблення стандартних бенчмарків і забезпечення відтворюваності результатів. Загалом ця праця чітко підсумувала сучасні підходи та труднощі, з якими стикаються дослідники під час впровадження машинного навчання моделі автоматизованого тестування ПЗ і вказала напрями, над якими ще варто працювати у найближчій перспективі.

У праці [6] наведено фреймворк SDC-Scissor, робота якого поєднує аналіз історичних даних про помилки з моделями на підставі нейронних мереж. Такий спосіб істотно зменшує кількість непотрібних тестів й увага зосереджується на найважливіших сценаріях, які видозмінюються з великою періодичністю. Так, автори зазначають: "SDC-Scissor перевершує випадковий базовий підхід у виборі небезпечних тестів за всіма варіантами складу тестового набору" [6, с. 24]. Дослідники наголошують на тому, що поєднання логів з попередніх запусків і машинного навчання моделей є надзвичайно ефективним, адже дає змогу краще адаптувати процес тестування ПЗ до змін програмного середовища.

Отже, проведений аналіз останніх досліджень та публікацій підтверджує вагому зацікавленість дослідників у застосуванні машинного навчання моделей для автоматизованого тестування ПЗ. Проте, попри наявність численних підходів і алгоритмів, питання комплексної систематизації, порівняльного оцінювання їх ефективності та визначення практичних рекомендацій з інтеграції в CI/CD-процеси залишається недостатньо дослідженим. Саме тому проведення цього дослідження є актуальним і своєчасним, адже воно дає змогу узагальнити наявні напрацювання та окреслити напрями їх практичного впровадження у сучасних проєктах.

**Матеріали та методи дослідження**. Під час виконання дослідження було опрацьовано різноманітні наукові напрацювання, з використання методів машинного навчання моделей автоматизованого тестування ПЗ. Основну увагу зосереджено на сучасних підходах до автоматичного генерування тестів, вибору релевантних сценаріїв тестування ПЗ, впорядкування тестових наборів та, як наслідок, скорочення тривалості процесу тестування ПЗ.

Для збирання матеріалу застосовано метод контент-аналізу наукових публікацій, опублікованих у період з 2012 по 2025 р. для обрання та опису найновіших робіт та думок дослідників. Усі праці зосереджені на застосуванні алгоритмів глибокого навчання, Reinforcement Learning та нейронних мереж. Окрім цього, розглядали праці про гібридні рішення, які поєднують ML із метаевристичними підходами.

Метод аналізу та синтезу використано для узагальнення отриманої інформації, виділення головних переваг та своєрідних обмежень кожного підходу. Окрему увагу приділено низці певних викликів: обмежений доступ до якісних навчальних даних, складність інтеграції ML у CI/CD-процеси та недостатня кількість прикладних досліджень за промислових умов.

Для формування висновків використовували приклади з реальних застосувань ML-моделей, зокрема – проєктів, де машинне навчання застосовують для вибору критичних тест-кейсів. Отже, дослідження базується на теоретичних джерелах і практиці. Тому, можна стверджувати, що наведене дослідження дає комплексне уявлення про сучасне застосування методів машинного навчання моделей автоматизованого тестування ПЗ, його можливості, перспективи та актуальні проблеми.

## Результати дослідження та їх обговорення / Research results and their discussion

Завдяки проведеному контент-аналізу наукових публікацій зі застосування машинного навчання моделей автоматизованого тестування ПЗ за 2012–2025 рр., вдалося виявити ключові підходи, які демонструють найвищу ефективність. Так, варто виділити [22]:

- глибоке навчання моделі (англ. *Deep Learning*);
- навчання з підкріпленням (англ. *Reinforcement Learning*, RL);
- гібридні підходи (англ. ML + метаевристики);
- класифікація та кластеризація тестових сценаріїв і логів;
- аналіз логів і телеметрії;
- онлайн-навчання (англ. *Online Learning*).

Загальну архітектуру інтеграції методів машинного навчання моделей в CI/CD-процеси автоматизованого тестування ПЗ наведено на рисунку.

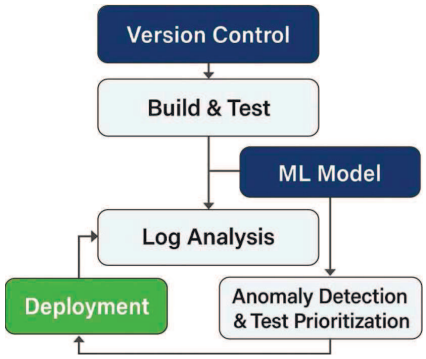


**Рисунок.** Архітектура інтеграції ML у CI/CD-процеси автоматизованого тестування ПЗ / Architecture of ML Integration into CI/CD Processes for Automated Software Testing

На схемі продемонстровано послідовність інтеграції методів машинного навчання моделей у CI/CD-процеси, де ML-модель використовують для аналізу логів, виявлення аномалій та пріоритизації тестів для підвищення ефективності розгортання ПЗ.

***Глибоке навчання моделі в автоматизованому тестуванні ПЗ***. Глибоке навчання (англ. *Deep Learning*, DL) моделі має значний потенціал для покращення автоматизованого тестування ПЗ через свою здатність знаходити приховані закономірності в даних, автоматично створювати потрібні тести, додавати логіку перевірки та покращити охоплення коду тестами.

У роботі [12] наведено інструмент CubeTesterAI, який використовує мовну модель LLaMA-2 для автоматичного розроблення JUnit-тестів мовою Java з використанням Spring Boot. У проведеному дослідженні модель досягла значно вищого покриття коду порівняно з традиційними інструментами. Модель LLaMA-2 навчалася на великій кількості функціонально пов'язаних пар "функція – тест" та показала свої можливості для автоматичного генерування синтаксично коректних тестів. Також було підтверджено, що розроблені тести успішно компілюються та проходять виконання без помилок. Так, це свідчить про практичну придатність моделі для реального використання в CI/CD-середовищах. Наприклад, у цій роботі зазначено, що для проєкту MongoDB CRUD (12 методів) модель LLaMA-2 L70B згенерувала 38 тестів, з яких 29 успішно пройшли перевірку, забезпечивши 94 % покриття коду, тоді як моделі меншої потужності (L7B, L13B, L34B) досягали тільки 0–29 %

покриття. Це підтверджує практичну перевагу використання CubeTesterAI на базі LLaMA-2 для автоматизації процесу генерування тестів і підвищення надійності ПЗ.

Пріоритезація тестів – ще одна сфера, де DL-додатки досягають належної ефективності [25]. Модель DeepOrder, аналізуючи історичні логи CI/CD, удосконалює порядок виконання тестів для швидшого виявлення дефектів. Завдяки цьому підходу можна зменшити час на перевірку коду без втрати його якості, оскільки першими виконуються саме ті тести, які з найбільшою ймовірністю виявляють помилки. Глибоке навчання моделі системи дає змогу їй адаптуватися до змін у поведінці коду та на підставі попереднього досвіду робити точніші припущення щодо потенційних зон ризику в нових версіях ПЗ.

Також варто згадати сучасні дослідження [10, 30, 32], в яких глибоке навчання моделі використовують для виявлення аномалій ПЗ під час його тестування у середовищі безперервної інтеграції. У роботі [10] описано застосування моделі Autoencoder для аналізу продуктивності ПЗ. Завдяки такому підходу вдалося зафіксувати стандартні збої та виявити приховані проблеми (витоки пам'яті, які не завжди видно з результатів типу "тест пройдено/не пройдено"). Порівняння з іншими методами показало, що модель Autoencoder має вищу чутливість до аномалій ПЗ. Оскільки фахівці підтвердили частину виявлених відхилень, тому можна вважати, що такий підхід має істотну практичну цінність.

У сфері тестування самих DL-систем глибоке навчання моделей застосовують у двох основних підходах [32]: метаморфному та мутаційному тестуванні ПЗ. Так, у дослідженні [16] демонструють, як можна вводити дрібні зміни у тренувальні дані, код та нейронні мережі для оцінювання стійкості моделі до помилок. Такий підхід дає змогу визначити, наскільки тестовий набір здатен виявити штучно згенеровані дефекти. Метаморфне тестування ПЗ зі свого боку базується на трансформації вхідних даних та дає змогу перевірити, як реагує модель у ситуаціях, коли немає однозначно правильного результату [21]. Перспективним вважають також гібридний підхід до тестування ПЗ, наведений, зокрема, у DeepEvolution [7], де еволюційні алгоритми поєднуються з глибинним навчанням моделей для генерування складних, рідкісних сценаріїв тестування ПЗ (corner cases) та досягнення ширшого охоплення роботи нейронної мережі.

***Навчання з підкріпленням*** RL (англ. *Reinforcement Learning*). Ще одним перспективним напрямом застосування глибокого навчання моделі в автоматизованому тестуванні ПЗ є навчання з підкріпленням. Цей підхід дає змогу моделі системи самостійно вивчати ефективні стратегії тестування ПЗ шляхом взаємодії з програмним середовищем. У роботі [29] наведено фреймворк DeepRNG, який використовує алгоритми глибокого підкріплення для генерування тестових сценаріїв із поступовим покращенням результатів. Автори зазначають, що така система здатна адаптуватися до змін у поведінці коду. Наприклад, було показано, що використання різних агентів RL для тестування Cosmos SDK дало найкраще покриття коду $11.883^{\pm 0.018}$ % (CS-SR агент), що статистично значуще ($p = 0.032$) порівняно з іншими методами. Саме тому цей підхід є особливо ефективним для тестування великих і динамічних програмних систем. RL-моделі також успішно застосовують для автоматичного оновлення пріоритетів тестів після змін у програмному коді. Наприклад, метод Retecs використовує підхід із підкріпленням для вибору та пріоритизації тест-кейсів у CI-середовищі [26]. Модель навчається на підставі історії виконання (час запуску, результати, провали тощо) та адаптує чергу тестів для того, щоб зменшити тривалість реакції на помилки.

***Гібридні підходи*** (*ML + метаевристики*). Ще одним перспективним напрямом удосконалення автоматизованого тестування ПЗ є застосування гібридних підходів, які поєднують методи машинного навчання моделей з метаевристичними алгоритмами. Такий підхід дає змогу одночасно враховувати історичні закономірності поведінки програмної системи, виявлені ML-моделями, та адаптивні механізми пошуку, притаманні метаевристикам. На практиці це забезпечує гнучкіше формування тестових сценаріїв, які здатні реагувати на складні зміни у програмному коді та середовищі виконання. Наприклад, у дослідженні [6] розглянуто підхід, який поєднує аналіз логів з нейронними мережами для визначення пріоритетних тестів у складних сценаріях, зокрема – для автономного ПЗ автомобілів. У цьому разі застосування гібридної моделі дало змогу зменшити кількість надлишкових тестів і зосередити увагу саме на критичних зонах ризику.

Ще один приклад подано у роботі [4], де автори досліджують можливості алгоритму мурашиної колонії (англ. *Ant Colony Optimization*) та алгоритму світлячків (англ. *Firefly Algorithm*) у завданні пріоритизації тест-кейсів. У цьому підході метаевристики використовують для ефективного впорядкування тестів залежно від обраних критеріїв, зокрема – покриття помилок і тривалості виконання. Такий метод дає змогу ефективно скоротити обсяг процесу тестування ПЗ без втрати його якості, що є особливо актуальним для складних систем із великою кількістю функціональних залежностей, які потребують багаторівневої перевірки. У дослідженні [3] також запропоновано гібридну стратегію, яка поєднує методи машинного навчання моделей з метаевристичним алгоритмом планування Horse Herd. Спочатку ML-модель використовують для виявлення потенційно дефектних ділянок коду, після чого оптимізаційний алгоритм відбору формує ефективний набір тестів, спрямований на охоплення найуразливіших зон. Автори вказують на те, що така комбінація методів тестування ПЗ дає змогу значно підвищити точність виявлення помилок й істотно скоротити час на генерування релевантних тестових сценаріїв.

4. *Класифікація та кластеризація.* Методи класифікації та кластеризації тестових сценаріїв і логів також активно застосовують в автоматизованому тестуванні ПЗ для аналізу великої кількості тестових сценаріїв, журналів виконання та програмних артефактів. Зокрема, класифікаційні алгоритми, такі як SVM, Random Forest чи нейронні мережі, дають змогу автоматично визначати, чи є конкретний тест-кейс релевантним, або прогнозувати ймовірність виявлення ним дефекту у ПЗ. Це особливо корисно за умов, коли потрібно швидко обрати підмножину тестів із великого їх набору. У дослідженні [33] показано, що поєднання класифікації тестових сценаріїв на підставі історичних даних з інформацією про частоту змін у програмному коді дає змогу зменшити кількість нерелевантних перевірок на етапі регресійного тестування ПЗ.

Кластеризацію, водночас, застосовують для групування подібних тест-кейсів або поведінкових шаблонів ПЗ. Завдяки цьому можна виявити дублікати, покращити охоплення ПЗ тестовими сценаріями або вдосконалити черговість запуску тестів. Наприклад, у роботі [24] було проведене комплексне порівняння алгоритмів кластеризації для виявлення аутлаєрів у різноманітних наборах даних, зокрема, у контексті логів CI/CD . Отримані результати показали, що класичні методи кластеризації (наприклад, k-means та DBSCAN) можуть ефективно групувати дані та самостійно виділяти аномальні шаблони ПЗ без потреби в ручному маркуванні. Додаткове дослідження [14] сфокусовано на розробленні техніки пріоритизації й відбору тест-кейсів для Agile-регресійного тестування ПЗ на підставі кластеризації. Автори пропонують спочатку згрупувати тест-кейси за подібністю до функціональних змін у спринтах, після чого виконати ранжування всередині кластерів. У такий спосіб процес тестування ПЗ адаптується автоматично до динаміки етапів проєкту, зменшуючи надлишковість перевірок і зберігаючи релевантність тестових сценаріїв.

Можна зробити висновок, що обидва підходи – класифікація й кластеризація – значно збільшують масштабованість та адаптивність процесу тестування ПЗ, що, водночас, є особливо важливо за умов сучасних CI/CD-середовищ.

***Аналіз логів і телеметрії***. Важливу роль під час тестування ПЗ відіграє також аналіз логів і телеметрії, оскільки ці джерела містять критично важливу інформацію про поведінку ПЗ в реальному часі. Великі програмні системи генерують значні обсяги логів, які складно обробити вручну, тому дедалі частіше застосовують ML-підходи до автоматизованого виявлення аномалій ПЗ, прогнозування його збоїв і встановлення кореневих причин помилок. Наприклад, у дослідженні [13] зазначено, що сучасні фреймворки лог-аналізу містять модулі для попереднього парсингу, векторизації повідомлень та їх подальшої кластеризації, що дає змогу ефективно виявляти відхилення у поведінці програмної системи. Аналіз телеметрії (метрики, трасування, події) також важливий: у роботі [1] показано, що застосування LSTM-Autoencoder дає змогу ефективно виявляти аномалії ПЗ у потоках даних серверної інфраструктури, забезпечуючи швидке його реагування. Ще одна робота [30] продемонструвала, що використання методів розкладу часових серій (наприклад, VMD) у поєднанні з LSTM значно скорочує затримку виявлення збоїв роботи ПЗ у реальному часі у середовищах із регулярним повторенням шаблонів даних. Це сприяє автоматизації процесу діагностики, підвищенню надійності CI/CD-процесів і загальному покращенню якості ПЗ.

***Онлайн-навчання*** (англ. *Online Learning*). Онлайн-навчання дає змогу машинному навчанню моделі безперервно адаптуватися до нових даних у реальному часі, що важливо за умов постійних змін CI/CD-середовищ і концептуального дрейфу даних. У контексті автоматизованого тестування ПЗ та моніторингу його роботи це означає, що модель може обробляти телеметричні потоки "на ходу" без потреби повторного аналізу вручну на кожному етапі. Наприклад, у дослідженні [31] наведено OML-AD – онлайн-підхід до детекції аномалій у часових рядах, який успішно виявляє збої роботи ПЗ під час концептуальних змін у даних. У роботах [1, 30] запропоновано адаптивний підхід на підставі LSTM-Autoencoder, який здатний динамічно виявляти аномалії ПЗ без ручного його налаштування після кожного оновлення сервера . Також ефективним є використання сучасних підходів контрольованого інкрементального навчання для запобігання "забуття" старих знань та підтримання стабільної продуктивності моделей під час поступового оновлення архітектури тест-системи.

Отже, результати контент-аналізу підтверджують, що застосування машинного навчання моделей автоматизованого тестування ПЗ розширює технічні його можливості та забезпечує адаптивність, масштабованість та гнучкість процесів. Кожен із розглянутих підходів вирішує окремі практичні завдання, що дає змогу дещо ефективніше реагувати на зміни у програмному середовищі. Отримані спостереження створюють основу для подальших досліджень і вдосконалення методів на стику ML і QA-практик. Для наочності у таблиці наведено узагальнене порівняння ефективності застосування різних підходів до машинного навчання моделі автоматизованого тестування ПЗ. У таблиці наведено приклади робіт, основні метрики його оцінювання та досягнуті результати.

**Таблиця.** Порівняння ефективності застосування різних підходів до машинного навчання моделі автоматизованого тестування ПЗ / Comparison of the Effectiveness of Different Machine Learning Approaches in Software Automated Testing Models

| Підхід | Приклад роботи | Метрика | Результат |
|---|---|---|---|
| Deep Learning | [12] CubeTesterAI | Coverage (покриття коду) | Досягнуто до 94 % покриття коду при генерування JUnit-тестів для MongoDB CRUD |
| Reinforcement Learning | [29] DeepRNG | Adaptability (адаптивність) | Покращене генерування тестових сценаріїв |
| Hybrid (ML+Metaheuristic) | [3] Horse Herd + ML | Time Reduction (скорочення часу) | Зниження часу генерування тестів |
| Clustering | [14] Claster-based prioritization | Precision (точність виявлення помилок) | Швидша пріоритизація тестів |

Наведені дані таблиці демонструють, що кожен із розглянутих підходів має свої переваги та сфери ефективного їх застосування в автоматизованому тестуванні ПЗ. Це підтверджує, що інтеграція методів машинного навчання моделей забезпечує значне підвищення якості ПЗ, швидкості та надійності його тестування у сучасних CI/CD-середовищах.

**Обговорення результатів дослідження**. У роботі [5] розглянуто GRU-based deep learning модель для пріоритезації тест-кейсів у CI/CD-середовищах. Автори застосували Gated Recurrent Unit для ідентифікації найрелевантніших тестів на підставі історичних даних про збої роботи ПЗ, враховуючи їхню тривалість, зміни станів і періоди між останніми запусками. Модель показала значне скорочення тривалості виконання (Total Algorithm Running Time – 0.02 s та 0.01 s) та високу fault detection ефективність (AFPD = 0.77; 0.70) порівняно з іншими підходами. Це підтверджує доцільність інтеграції методів глибинного навчання для удосконалення процесу тестування ПЗ та узгоджується з результатами, отриманими у нашому дослідженні.

У роботі [11] наведено новий алгоритм оптимізації QLOBL_INFO, який поєднує навчання з підкріпленням (RL) та opposition-based learning (OBL) для покращення

вибору стратегій удосконалення процесу тестування ПЗ у складних завданнях. Автори застосували Q-learning для динамічного відбору найефективніших OBL-варіантів, що дало змогу досягти високих результатів за розв'язання задач вибору ознак та обмежених оптимізаційних задач на наборах CEC2022. Це демонструє потенціал RL-механізмів для посилення ефективності удосконалених процесів, що узгоджується з висновками цього дослідження щодо доцільності інтеграції ML у завдання автоматизованого тестування ПЗ.

У праці [18] було запропоновано алгоритм CDEA-DQN, заснований на Deep Q Network, для вирішення завдання впорядкування розміщення сенсорів у мультисенсорних позиціювальних системах (MSPS) за умов складних робочих середовищ. Автори розглянули стратегії винагороди та оновлення, що забезпечують адаптивне відтворення, швидку збіжність та високу точність. Результати проведених експериментів на 44 тестових випадках підтвердили ефективність алгоритму порівняно з іншими наявними методами. Це демонструє потенціал використання глибинного навчання з підкріпленням для складних оптимізаційних задач, що узгоджується з висновками цього дослідження про доцільність застосування ML-підходів для підвищення продуктивності та якості рішень.

У роботі [23] розглянуто використання методів штучного інтелекту для підвищення безпеки CI/CD-процесів шляхом виявлення аномалій у мережевому трафіку. Автори застосували комбінацію Convolutional Neural Network (CNN) та Long Short-Term Memory (LSTM) для ідентифікації нетипових патернів, що можуть свідчити про DDoS-атаки, боти чи інші кіберзагрози. Модель досягла належної точності (98.69 % та 98.30 %) під час тестування на наборах CSE-CIC-IDS2018 та CSE-CIC-IDS2017. Отримані результати демонструють ефективність глибинного навчання у завданнях безперервної інтеграції, що узгоджується з результатами цього дослідження щодо доцільності інтеграції ML-підходів у CI/CD-процесах для покращення якості ПЗ та надійності його роботи.

У дослідженні [15] було проаналізовано вплив гіперпараметрів на ефективність вирішення завдань пріоритизації тест-кейсів (TCP) у CI/CD-середовищах. Автори виконали експериментальний аналіз 885 комбінацій гіперпараметрів для машинного навчання чотирьох моделей на великих реальних наборах даних. Унаслідок, середня продуктивність моделей зросла на 15 % порівняно зі стандартними налаштуваннями, а досягнуті показники precision (max = 1), recall (max = 0.9633) та F1-score (max = 0.9662) підтверджують істотне їх покращення. Це демонструє безпосередній вплив удосконалення параметрів на якість пріоритизації тестів, що узгоджується з висновками нашого дослідження щодо необхідності коректного налаштування ML-моделей для підвищення результативності автоматизованого тестування ПЗ.

У праці [8] наведено метод DeepREST – інноваційний підхід black-box тестування REST API на підставі глибинного навчання з підкріпленням. Модель використовує curiosity-driven learning для вивчення послідовності операцій та визначення належних стратегій тестування ПЗ, що допомагає досягати високого покриття тестами та ефективного виявлення дефектів. Це підтверджує доцільність використання RL-алгоритмів у завданнях автоматизованого тестування ПЗ, що узгоджується з висновками цього дослідження про ефективність ML-підходів для покращення надійності програмних систем.

Ще одним прикладом є дослідження [20], у якому запропоновано гібридний підхід до удосконалення Target Set Selection задач на підставі поєднання deep learning, Q-learning та ant colony optimization. Автори продемонстрували, що використання інформації, отриманої за допомогою глибинного навчання тестових моделей у поєднанні з Q-learning і традиційними феромонними механізмами, забезпечує істотне підвищення ефективності автоматизованого тестування ПЗ порівняно з класичним підходом. Це підтверджує, що інтеграція ML-моделей і метаевристик, як показано в нашому дослідженні, є перспективною для вирішення складних оптимізаційних завдань, зокрема й з автоматизованим тестуванням ПЗ.

У дослідженні [17] використано фреймворк RESTest для автоматичного генерування та виконання тест-кейсів RESTful API за промислових умов. Автори провели експеримент на 13 API упродовж 15 днів безперервного тестування, виконавши понад мільйон тестів і виявивши близько 390 тис. збоїв, з яких 254 були підтверджені як баги. Це вказує на потенціал автоматизованого тестування API для значного підвищення якості ПЗ та узгоджується з результатами нашого дослідження, де доведено ефективність ML-підходів для масштабного генеративного тестування у CI/CD-середовищах.

Ще одна актуальна праця [27] стосується автоматичної адаптації гіперпараметрів у метаевристичних алгоритмах на підставі reinforcement learning. Автори запропонували універсальний RL-фреймворк для навчання стратегій зміни його параметрів, продемонструвавши ефективність на прикладі CMA-ES (англ. *Covariance Matrix Adaptation Evolution Strategies*) та DE (англ. *Differential Evolution*). Експерименти на 46 тестових функціях показали конкурентоспроможність запропонованих політик у більшості випадків. Це підтверджує перспективність RL-підходів для автоматизації складних процесів, зокрема – у завданнях удосконалення процесу тестування ПЗ, що узгоджується з висновками цієї роботи.

Отже, проведене обговорення результатів дослідження показало, що хоча існує низка успішних рішень з використанням ML та RL у суміжних завданнях удосконалення процесу тестування ПЗ, пріоритизації тест-кейсів і генерування тестів, жодна з проаналізованих робіт не пропонує комплексного підходу, який одночасно враховує пріоритизацію, генерування та удосконалення тестів у CI/CD-середовищах із інтеграцією різних ML-моделей та метаевристик. Саме це вказує на доцільність проведеного дослідження та його наукову новизну.

Отже, внаслідок виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

*Наукова новизна отриманих результатів дослідження* – отримала подальший розвиток методика комплексного контент-аналізу особливостей застосування методів машинного навчання моделей, яка, на відміну від наявної, враховує глибинне їх навчання, навчання з підкріпленням та гібридні ML+метаевристичні підходи до автоматизованого тестування ПЗ у CI/CD-середовищах із виділенням їхньої ефективності, практичної реалізації та ключових метрик.

*Практична значущість результатів дослідження* – отримані результати можна використати під час впровадження ML-підходів у процеси автоматизованого тестування ПЗ, зокрема – для вибору належних методів генерування та пріоритизації тестів, скорочення тривалості виконання CI/CD-процесів і підвищення якості продукту проєкту завдяки швидкому виявленню дефектів і аномалій ПЗ.

## Висновки / Conclusions

Проаналізовано можливості застосування машинного навчання моделей для підвищення ефективності автоматизованого тестування ПЗ, що дає змогу отримати максимум інформації про якість його роботи в межах виділених часових ресурсів проєкту, зменшити кількість як дрібних, так і більших дефектів, а також забезпечує стабільнішу роботу продукту проєкту. За результатами виконаного дослідження можна зробити такі основні висновки.

1. Результати аналізу новітніх підходів до валідації підтвердили високу ефективність інтеграції методів машинного навчання моделей автоматизованого тестування ПЗ. Здійснений контент-аналіз наукових джерел дав змогу виокремити ключові напрями, які демонструють практичну цінність: глибоке навчання моделі, навчання з підкріпленням, гібридні ML+метаевристичні стратегії, класифікація та кластеризація, аналіз логів і телеметрії та онлайн-навчання.
2. Встановлено, що застосування DL-архітектур та RL-фреймворків дає змогу автоматизувати процес генерування тестів, забезпечити їх адаптивність до змін коду й підвищити якість виявлення помилок. Використання історичних логів і телеметрії сприяє кращій орієнтації тестів на критичні тестові сценарії та дає змогу оперативно реагувати на аномалії ПЗ. Гібридні підходи, які поєднують можливості машинного навчання моделі й алгоритмів покращення тестових сценаріїв, забезпечують зменшення надлишкових перевірок та пріоритизацію виконання тестів. Методи класифікації та кластеризації сценаріїв тестування ПЗ допомагають ефективно працювати з великими обсягами тестових даних, а онлайн-навчання – динамічно оновлювати моделі без втрати продуктивності.
3. Дослідження процесу інтеграції машинного навчання моделі у процеси автоматизованого тестування ПЗ засвідчило наявність низки викликів: складність масштабування рішень у CI/CD-середовищах, обмежений доступ до якісних навчальних даних, труднощі із забезпеченням відтворюваності результатів і стандартизації процесу оцінювання якості ПЗ з погляду покриття тестами.
4. Проведене дослідження засвідчило наявність низки викликів: складність масштабування ML-рішень у CI/CD-середовищах, обмежений доступ до якісних навчальних даних, труднощі із забезпеченням відтворюваності результатів і стандартизації процесу оцінювання моделей автоматизованого тестування ПЗ.
5. На підставі отриманих результатів сформульовано доцільність проведення подальших етапів дослідження стосовно розроблення універсальних фреймворків, здатних ефективно працювати за умов постійних змін програмного середовища, та створення практичних рекомендацій щодо впровадження ML-рішень у промислові CI/CD-процеси для підвищення ефективності та надійності ПЗ.

## References


1. Abdennebi, A., Tuncay, A., Yilmaz, C., Koyuncu, A., & Gungor, O. (2023, May). LSTM-AE for anomaly detection on multivariate telemetry data. In *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications* (*SERA*) (pp. 90–97). IEEE. https://doi.org/10.1109/SERA57763.2023.10197673
2. Abo-eleneen, A., Palliyali, A., & Catal, C. (2023). The role of Reinforcement Learning in software testing. *Information and Software Technology*, *164*, article ID 107325. https://doi.org/10.1016/j.infsof.2023.107325
3. Arasteh, B., Arasteh, K., & Ghaffari, A. (2025). An automatic software test-generation method to discover the faults using fusion of machine learning and horse herd algorithm. *The Journal of Supercomputing*, *81*(5), 1–36. https://doi.org/10.1007/s11227-025-07219-5
4. Ariffin, M. A., Ibrahim, R., Ibrahim, I. S., & Wahab, J. A. (2022). Test cases prioritization using ant colony optimization and firefly algorithm. International Journal of Engineering Trends and Technology, 70(3), 22–28. https://doi.org/10.14445/22315381/IJETT-V70I3P203
5. Behera, A., & Acharya, A. A. (2025). An Effective GRU-Based Deep Learning Method for Test Case Prioritization in Continuous Integration Testing. *Procedia Computer Science*, 258, 4070–4083. https://doi.org/10.1016/j.procs.2025.04.658
6. Birchler, C., Khatiri, S., Bosshard, B., Gambi, A., & Panichella, S. (2023). Machine learning-based test selection for simulation-based testing of self-driving cars software. *Empirical Software Engineering*, *28*(3), article ID 71. https://doi.org/10.1007/s10664-023-10286-y
7. Braiek, H. B., & Khomh, F. (2019, September). Deepevolution: A search-based testing approach for deep neural networks. In *2019 IEEE International Conference on Software Maintenance and Evolution* (*ICSME*) (pp. 454–458). IEEE. https://doi.org/10.1109/ICSME.2019.00078
8. Corradini, D., Montolli, Z., Pasqua, M., & Ceccato, M. (2024, October). DeepREST: Automated Test Case Generation for REST APIs Exploiting Deep Reinforcement Learning. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (pp. 1383–1394). https://doi.org/10.1145/3691620.3695511
9. Fontes, A., & Gay, G. (2023). The integration of machine learning into automated test generation: A systematic mapping study. *Software Testing, Verification and Reliability*, *33*(4), article ID e1845. https://doi.org/10.1002/stvr.1845
10. Gerber, D., Meitz, L., Rosenbauer, L., & Hähner, J. (2024). Unsupervised anomaly detection in continuous integration pipelines. URL: https://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/117136
11. Gölcük, İ., Ozsoydan, F. B., & Durmaz, E. D. (2025). Reinforcement and opposition-based learning enhanced weighted mean of vectors algorithm for global optimization and feature selection. *Knowledge-Based Systems*, article ID 113626. https://doi.org/10.1016/j.knosys.2025.113626
12. Gorla, D., Kumar, S., Lorenzini, P. N. R., & Alipourfaz, A. (2025, March). CubeTesterAI: Automated JUnit Test Generation Using the LLaMA Model. In *2025 IEEE Conference on Software Testing, Verification and Validation* (*ICST*) (pp. 565–576). IEEE. https://arxiv.org/pdf/2504.15286
13. He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. *ACM computing surveys* (*CSUR*), *54*(6), 1–37. https://doi.org/10.1145/3460345
14. Kandil, P., Moussa, S., & Badr, N. (2017). Cluster-based test cases prioritization and selection technique for agile regression testing. *Journal of Software: Evolution and Process*, *29*(6), article ID e1794. https://doi.org/10.1002/smr.1794
15. Khan, M. A., Azim, A., Liscano, R., Smith, K., Chang, Y. K., Tauseef, Q., & Seferi, G. (2024, April). Machine learning-based test case prioritization using hyperparameter optimization. In *Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test* (AST 2024) (pp. 125–135). https://doi.org/10.1145/3644032.3644467

16. Ma, L., Zhang, F., Sun, J., Xue, M., Li, B., Juefei-Xu, F., & Wang, Y. (2018, October). Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th international symposium on software reliability engineering* (*ISSRE*) (pp. 100–111). IEEE. https://doi.org/10.48550/arXiv.1805.05206
17. Martin-Lopez, A., Segura, S., & Ruiz-Cortés, A. (2022, November). Online testing of RESTful APIs: Promises and challenges. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 408–420). https://doi.org/10.1145/3540250.3549144
18. Ning, Y., Bai, Z., Wei, J., Suganthan, P. N., Xing, L., Wang, J., & Song, Y. (2025). A meta-heuristic algorithm combined with deep reinforcement learning for multi-sensor positioning layout problem in complex environment. *Expert Systems with Applications*, 261, article ID 125555. https://doi.org/10.1016/j.eswa.2024.125555
19. Rahman, S. M. M., & Eisty, N. U. (2024). Introducing ensemble machine learning algorithms for automatic test case generation using learning based testing. *arXiv preprint arXiv:2409.04651*. https://doi.org/10.48550/arXiv.2409.04651
20. Ramírez Sánchez, J. E., Chacón Sartori, C., & Blum, C. (2023, July). Q-Learning ant colony optimization supported by deep learning for target set selection. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 357–366). https://doi.org/10.1145/3583131.3590396
21. Rehman, F. U., & Srinivasan, M. (2023, July). Metamorphic testing for machine learning: Applicability, challenges, and research opportunities. In *2023 IEEE International Conference On Artificial Intelligence Testing* (*AITest*) (pp. 34–39). IEEE. https://doi.org/10.1109/AITest58265.2023.00014
22. Salam, M. A., Abdel-Fattah, M., & Moemen, A. A. (2022). A survey on software testing automation using machine learning techniques. *International Journal of Computer Applications*, *183*(51), 12–19. https://doi.org/10.5120/ijca2022921919
23. Saleh, S. M., Sayem, I. M., Madhavji, N., & Steinbacher, J. (2024, November). Advancing Software Security and Reliability in Cloud Platforms through AI-based Anomaly Detection. In *Proceedings of the 2024 on Cloud Computing Security Workshop* (pp. 43–52). https://doi.org/10.1145/3689938.3694779
24. Sánchez Vinces, B. V., Schubert, E., Zimek, A., & Cordeiro, R. L. (2025). A comparative evaluation of clustering-based outlier detection. *Data Mining and Knowledge Discovery*, *39*(2), article ID 13. https://doi.org/10.1007/s10618-024-01086-z
25. Shankar, R., & Sridhar, D. (2024). An Improved Deep Learning Based Test Case Prioritization Using Deep Reinforcement Learning. *International Journal of Intelligent Engineering & Systems*, *17*(1), 771–782. https://doi.org/10.22266/ijies2024.0229.64
26. Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2017, July). Reinforcement learning for automatic test case prioritization and selection in continuous integration. In *Proceedings of the 26th ACM SIGSOFT international symposium on software testing and analysis* (pp. 12–22). https://doi.org/10.1145/3092703.3092709
27. Tessari, M., & Iacca, G. (2022, July). Reinforcement learning based adaptive metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 1854–1861). https://doi.org/10.1145/3520304.3533983
28. Tsai, C. Y., & Taylor, G. W. (2022). Deeprng: Towards deep reinforcement learning-assisted generative testing of software. *arXiv preprint arXiv: 2201.12602*. https://doi.org/10.48550/arXiv.2201.12602
29. Tsai, C. Y., & Taylor, G. W. (2022). Deeprng: Towards deep reinforcement learning-assisted generative testing of software. *arXiv preprint arXiv: 2201.12602*. https://doi.org/10.48550/arXiv.2201.12602
30. Vajda, D. L., Do, T. V., Bérczes, T., & Farkas, K. (2024). Machine learning-based real-time anomaly detection using data pre-processing in the telemetry of server farms. *Scientific Reports*, *14*(1), article ID 23288. https://doi.org/10.1038/s41598-024-72982-z
31. Wette, S., & Heinrichs, F. (2024). OML-AD: Online Machine Learning for Anomaly Detection in Time Series Data. *arXiv preprint arXiv: 2409.09742*. https://doi.org/10.48550/arXiv.2409.09742
32. Xiao, D., Liu, Z., Yuan, Y., Pang, Q., & Wang, S. (2022). Metamorphic testing of deep learning compilers. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, *6*(1), 1–28. https://doi.org/10.1145/3508035
33. 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
34. Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, *48*(1), 1–36. https://doi.org/10.1109/TSE.2019.2962027

***O. I. Torskyi, Yu. I. Hrytsiuk***

*Lviv Polytechnic National University, Lviv, Ukraine*

## APPLICATION OF MACHINE LEARNING TO ENHANCE THE EFFICIENCY OF AUTOMATED SOFTWARE TESTING

This study investigates the integration of machine learning (ML) techniques into automated software testing, with a primary focus on modern approaches. Through a comprehensive content analysis of peer-reviewed publications from 2012 to 2025, the research identifies key methods such as deep learning architectures, reinforcement learning (including Q-learning), and hybrid models that combine ML with metaheuristic strategies. Special attention is given to solutions that optimize test case generation and prioritize critical tests. As a result, it was revealed that leveraging historical execution logs, telemetry data, and continuous integration (CI) artifacts significantly enhances the relevance of generated tests and shortens their execution time. For example, systems like SDC Scissor, which utilize neural networks, demonstrate superior performance in identifying unsafe test cases compared to random baselines. In this way, the practical benefits of ML-driven test selection were higlighted. Moreover, the study outlines that supervised learning techniques, particularly those based on neural networks, are increasingly adopted alongside semi-supervised and unsupervised models to adapt testing processes to the dynamic nature of modern software systems. Nevertheless, several pressing challenges remain unresolved, including the limited availability of labeled training datasets, difficulties in model retraining, and integration scalability within CI/CD pipelines. The lack of standardized evaluation benchmarks and reproducibility standards continues to hinder progress in the field. The findings emphasize the growing potential of ML in transforming software testing practices. Additionally, the research highlights the importance of integrating anomaly detection mechanisms for real-time error identification during testing cycles. Online learning approaches are emerging as a promising solution for adapting to continuous data streams and addressing concept drift. The role of classification and clustering is further underscored, especially in prioritizing regression test suites and grouping similar execution patterns. Furthermore, deep generative models like autoencoders are increasingly being applied to detect hidden faults that evade traditional rule-based testing. Overall, the study underscores that ML is no longer experimental but an evolving standard in the pursuit of efficient and intelligent software quality assurance.

**Keywords:** deep learning; reinforcement learning; scenario prioritization; CI/CD processes; hybrid approaches.