УДК 004.8:004.05
DOI https://doi.org/10.32782/2663-5941/2025.2.2/15

Лебедєв А. В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Федорова Н. В.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МУТАЦІЙНИЙ АНАЛІЗ ТА ПРОБЛЕМИ ЙОГО ВИКОРИСТАННЯ ДЛЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

У статті розглянуто особливості застосування мутаційного аналізу до систем машинного навчання (МН). Проведений аналіз засвідчив, що класичне тестування мутаціями не може бути безпосередньо використане до систем з МН без відповідної адаптації. У статті детально описуються основні проблеми, які виникають під час мутаційного аналізу у системах МН, зокрема, визначення поняття «вихідного коду», розробка ефективних та реалістичних операторів мутації, оптимізація швидкодії процесу тестування і необхідність вдосконалення методів оцінки мутаційного покриття. Крім того, стаття розглядає останні дослідження мутаційного аналізу в контексті систем з МН та проводить їх критичний аналіз.

У ході дослідження запропоновано низку рекомендацій для дослідників та розробників програмного забезпечення, які дозволять не лише інтегрувати тестування мутаціями до моделей МН, а й зберегти основоположні концепції мутаційного аналізу. Наприклад, обґрунтовано необхідність чіткого розмежування між вихідним кодом і тестовими наборами, створення спеціалізованих операторів мутації та розробка комбінованих метрик для оцінки тестового покриття. Щобільше, розглянуто способи оптимізації обчислювальної складності мутаційного аналізу, які використовуються в наявних інструментах. Запропоновані рішення сприятимуть підвищенню якості тестування моделей МН та забезпеченню більш точної оцінки їхньої надійності.

Результати дослідження підкреслюють важливість адаптації класичних методів тестування до особливостей систем МН для підвищення їхньої ефективності та надійності. Висновки визначають напрямки подальших досліджень у цій сфері, зокрема, розробку нових реалістичних операторів мутації, оптимізацію швидкодії тестування та узгодження основних концептуальних аспектів мутаційного аналізу із принципами роботи моделей МН. Подальший розвиток методів мутаційного аналізу дозволить зробити тестування моделей МН більш надійним, ефективним і адаптивним до змінних умов реальних застосувань.

Ключові слова: *мутаційне тестування, оператори мутації, якість тестових наборів, машинне навчання, моделі машинного навчання, обробка великих масивів даних.*

Постановка проблеми. Моделі машинного навчання (МН), в особливості моделі глибокого навчання, зазнають активного розвитку та впровадження у низку критично-важливих галузей, які пов'язані з медициною, фінансами, енергетикою та військово-промисловим апаратом. Вони дозволяють автоматизувати складні процеси для прийняття рішень на основі аналізу великої кількості показників, які раніше вимагали втручання людини. Тому якість та надійність моделей МН є дуже важливими аспектами, оскільки навіть невелика помилка в результатах обчислень може впливати на життя людей.

Через комплексну природу моделей МН їх тестування характеризується високим рівнем складності. Охопити увесь спектр можливих проблем, які виникають в процесі розробки та навчання моделей, можна шляхом впровадження інтегрованого підходу на основі різних технік.

Тестування мутаціями або мутаційний аналіз – метод тестування програмного забезпечення, який гарно зарекомендував себе у традиційних системах. Однак він, як і інші традиційні методи тестування, потребують адаптації та переосмислення концепцій, адже процеси розробки та підтримки моделей МН суттєво відрізняються від аналогічних процесів у класичних системах.

Одними з ключових проблем мутаційного аналізу в контексті системи МН є визначення об'єкта тестування та розробка мутаційних операторів,

які дійсно імітують можливі людські помилки й впливають на роботу моделі. Крім того, існують проблеми, пов'язані з оптимізацією швидкості виконання аналізу та ефективним використанням обчислювальних ресурсів.

Тому, аналіз проблем використання тестування мутаціями в контексті моделей МН й розробка стратегічних напрямків їх вирішення є актуальними й необхідними завданнями, адже це має безпосередній вплив на забезпечення якості та стабільності систем у сучасних умовах експлуатації.

Аналіз останніх досліджень і публікацій. Питання мутаційного аналізу у традиційних системах було досліджено у роботах Ю. Цзя та М. Гарман [1, с. 650–653]. Пізніше тестування мутаціями було детально описано у роботі Ц. Чжу, А. Панічелли та Е. Зайдмана [2, с. 2–3].

Проте мутаційний аналіз для моделей МН є доволі новим напрямом досліджень, що обумовлює обмеженість публікацій у сфері. Однією з перших робіт у цій галузі є робота Л. Ма, Ф. Чжан, Цз. Сунь, М. Сю, Бо Лі та Ф. Цзюфей-Су [3, с. 100–101]. Автори вводять мутаційні оператори для моделей глибокого навчання, які передбачають модифікації як на рівні вихідного коду, так і на рівні навченої моделі

Майже у той самий час В. Шен, Цз. Ван та Чж. Чен [4, с. 110] представили методику тестування, яка заснована на мутаційних операторах виключно на рівні моделі.

Пізніше Ц. Ху, Л. Ма, С. Сі, Б. Ю, Ян Лю та Ц. Чжао [5, с. 1159] продовжили напрацювання проведені у роботі [3, с. 104] шляхом розширення мутаційного тестування для рекурентних нейронних мереж.

Однак підходи, описані у вищезгаданих роботах, зазнають значної критики зі сторони науковців. Наприклад, Г. Джахангірова та П. Тонелла [6, с. 74–75] розглядають неефективність концепцій, запропонованих у [3, с. 104], бо вони не враховують стохастичну природу процесу навчання, та пропонують альтернативний список мутаційних операторів, які дійсно мають вплив на моделі.

Крім того, Н. Гумбатова, Г. Джахангірова, Г. Бавота, В. Річчіо, А. Стокко та П. Тонелла [7, с. 1115] запропонували таксономію реальних несправностей у системах МН, виявивши, що багато з них ще не були виявлені під час тестування за допомогою існуючих інструментів. Також автори пропонують підходи, які можуть бути враховані у подальших роботах.

А. Панічелла та С. Ліем [8, с. 67–69] досліджують об'єкт тестування у моделях МН в процесі мутаційного аналізу. Вони описують проблеми сучасних методик та пропонують план для подолання виявлених проблем.

Аналіз останніх публікацій підтвердив зростаючий інтерес наукової спільноти до даної тематики. Щобільше, наскільки відомо автору статті, дана тема не є дослідженою у вітчизняній науковій літературі, що підтверджує актуальність роботи.

Постановка завдання. Метою статті є дослідження мутаційного аналізу та можливостей його використання до моделей МН, зокрема, до систем глибокого навчання. Для досягнення поставленої мети пропонується:

1. Здійснити аналіз наявних підходів та практик.
2. Ідентифікувати та детально описати проблеми.
3. Визначити список рекомендацій, які можуть допомогти у розв'язанні окреслених проблем.

Виклад основного матеріалу. Концепція мутаційного аналізу бере свій початок з 70-х років минулого століття, її згадування можна прослідкувати у роботах Р. А. ДеМілло, Р. Д. Ліптона, Ф. Г. Сейварда [9, с. 36] та Р. Г. Гамлета [11, с. 280]. Попри те, що ідея тестування мутаціями виникла досить давно, найбільшого свого розвитку вона зазнала лише в останні десятиліття у зв'язку зі збільшенням обчислювальних потужностей комп'ютерних систем.

Тестування мутаціями або мутаційний аналіз – це метод тестування програмного забезпечення, який полягає у внесенні невеликих змін у вихідний код системи задля перевірки ефективності тестового набору.

На рисунку 1 продемонстровано процес виконання мутаційного аналізу. Під час тестування утворюється велика кількість копій системи, кожна з яких мають одну невеличку зміну у вихідному коді – вони мають назву мутанти. Модифіковані версії програми генеруються за допомогою операторів мутації, які описують типові помилки розробників програмного забезпечення. Подальший запуск набору тестів для кожного з мутантів оцінює якість написаного тестового набору системи за певним алгоритмом: якщо тести пройшли на мутанті, то мутант вважається «вижившим», інакше – «вбитим». Якщо тестовий набір на мутанті пройшов без помилок, то це демонструє, що тести не покривають увесь спектр можливих проблем або перевіряють код поверхнево. У результаті аналізу отримують оцінку мутацій, яка є відношенням «вбитих» мутантів до їх загаль-

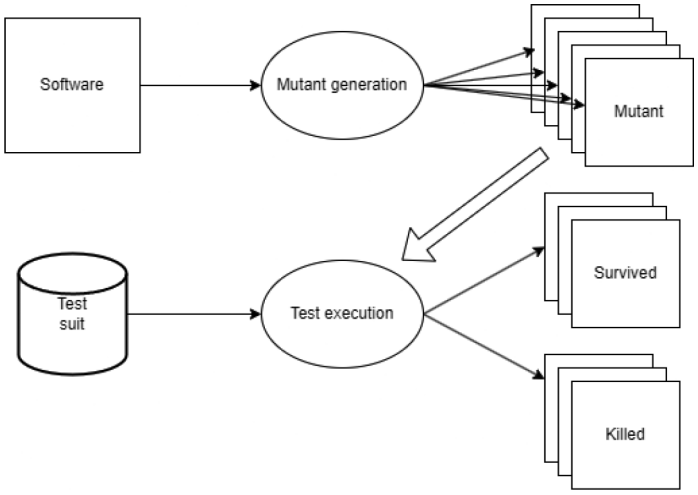

Рис. 1. Процес виконання мутаційного аналізу для традиційних систем

ної кількості. Високе значення оцінки свідчить про здатність тестового набору виявляти різноманітні потенційні помилки, а низький показник вказує на їхню низьку якість та наявність можливих прогалин у перевірці.

Як описують А. Панічелла та С. Ліем у своїй роботі [8, с. 67], мутаційний аналіз ґрунтується на двох гіпотезах:

1. Гіпотеза ефекту зв'язку (ГЕЗ) [9, с. 34] – описує залежність між різними за масштабом помилками. Основна ідея передбачає, що якщо тест здатний розпізнати маленьку зміну мутаційного оператора, як, наприклад, зміна знаку додавання () на віднімання (–), то з високою ймовірністю він виявить і більші зміни, які фактично є комбінацією декількох маленьких модифікацій.

2. Гіпотеза компетентного програміста (ГКП) [10, с. 5] – стверджує, що більшість програмістів є достатньо компетентними, щоб продукувати коректний або близький до коректного код. Під формулюванням «близький до коректного код» розуміють код, який може містити невеликі помилки.

Обидві гіпотези є взаємопов'язаними та критично важливими для обґрунтування ефективності мутаційного аналізу.

Важливо усвідомити, що в процесі мутаційного аналізу мутаціям піддається вихідний код системи, в той час, як об'єктом тестування є тестовий набір. Тобто фактично аналіз спрямований на перевірку якості тестового набору, що забезпечує оцінку коректності роботи вихідного коду.

Мутаційний аналіз також знаходить застосування у тестуванні систем МН, включаючи моделі глибокого навчання, як це демонструється у роботах [3, с. 102–104; 4, с. 108; 5, с. 1158]. Системи з МН – це системи, які мають одну чи більше моделей МН. Загалом процес МН складається з трьох основних етапів: навчання на тренувальних даних, верифікація на тестових даних та впровадження в експлуатацію. Як правило, тренувальна та тестова вибірки формуються на основі розбиття початкового набору даних у певному співвідношенні. При цьому, тестова вибірка містить дані, які раніше не були використані під час навчання, щоб оцінити роботу моделі в наближених до реальних умовах.

Попри довгу історію мутаційного тестування, його застосування для систем з МН розпочалося відносно нещодавно. Перші дослідження у цій сфері відзначають, що джерела помилок у моделях МН є значно різноманітнішими, ніж у традиційному програмному забезпеченні. Наприклад, Л. Ма та інші [3, с. 100] стверджують, що тренувальні дані, навчальна програма та навчена модель породжують найбільшу кількість помилок, тому автори пропонують мутаційні оператори для цих компонентів. Водночас В. Шен, Цз. Ван та Чж. Чен [4, с. 110] зосереджуються виключно на мутаціях навченої моделі.

Проте застосування мутаційного аналізу до МН створює низку концептуальних труднощів. Згідно з класичним визначенням мутаційного аналізу, мутаціям має піддаватися вихідний код системи, але його ідентифікація в контексті моделей МН є нетривіальною задачею.

На перший погляд, тренувальна програма, яка визначає структуру та гіперпараметри моделі, виглядає перспективним кандидатом, адже зміни в ній матимуть безпосередній вплив на модель. Однак тренувальні програми радше є специфікацією, на основі якої фреймворк МН будує модель. Якщо проводити аналогію з WEB-програмуванням, то тренувальна програма

є API-специфікацією, яку WEB-фреймворк може використати для генерації обробників запитів. Щобільше, самі автори [3, с. 107] у рамках проведеного експерименту щодо впливу мутацій на результат моделі зазначають, що мутаційні оператори рівня тренувальної програми породжують мутантів, які не переживають тестування через значну різницю у роботі моделі.

Тренувальні дані теж можуть зазнавати мутацій, проте, такі зміни насамперед впливають на логіку моделі, а не дають змогу оцінити повноту та якість тестового набору, що є ціллю мутаційного аналізу. Як стверджують А. Панічелла та С. Ліем [8, с. 66], навчання та оцінку моделі можна порівняти із двома послідовними Test-Driven Development (TDD) процесами. TDD – це практика розробки програмного забезпечення, у якій спочатку пишуть тестовий набір, а потім ітеративно розробляють код, допоки не пройдуть усі тести. Згідно з авторами, в процесі навчання тренувальні дані є тестовим набором, а ітеративний процес розробки – це процес навчання моделі до досягнення певної точності. У другому процесі тестові дані відповідають тестовому набору з TDD, а вихідним кодом також є навчена модель, але тут ітеративним процесом є не навчання, а зміна однієї моделі на іншу. Таким чином, тренувальні дані не мають піддаватися мутаціям, адже згідно з TDD аналогією вони не є «вихідним кодом» в обох випадках.

Навчену модель часто розглядають як найбільш доцільний об'єкт для внесення мутацій, адже вона найближча до поняття «вихідного коду» на етапі експлуатації. Однак постає питання, чи можуть зміни на цьому рівні дійсно імітувати помилки, притаманні розробникам? Адже зазвичай навчання моделі відбувається фреймворком за чітко визначеним алгоритмом, де немає місця випадковим модифікаціям, як, наприклад, додавання чи вилучення шару мережі.

У таблиці 1 наведено узагальнений порівняльний аналіз мутаційних змін на різних рівнях моделей МН, включаючи рівень тренувальних даних, тренувальної програми та навченої моделі. Таблиця містить приклади мутацій, їхні переваги та недоліки. Кожен рівень мутацій має свої сильні та слабкі сторони, що слід враховувати при виборі підходу до тестування моделей машинного навчання.

Окрім визначення вихідного коду, важливою концептуальною проблемою також є відповідність гіпотезам мутаційного аналізу. У традиційному випадку припускають, що ідентифікація дрібних помилок є маркерами, що забезпечують виявлення більших вразливостей за умови їх наявності. Утім, у контексті моделей МН вплив незначної модифікації не є передбачуваним і стабільним через стохастичну природу навчання моделі. Крім того, ГКП передбачає, що мутації імітують типові помилки розробників у вихідному коді. Якщо ж згенеровану фреймворком модель вважати "вихідним кодом", то яким чином мутації навченої моделі можуть імітувати помилки розробників? Очевидно, ці аспекти потребують детальнішого дослідження для досягнення узгодженості.

Також проблемним є питання швидкодії мутаційного аналізу як у традиційних системах, так і в системах МН. У класичному мутаційному аналізі застосовуються різні методи оптимізації, які дозволяють скорочувати час виконання аналізу та зменшувати обчислювальні витрати [1, с. 653–657]. Проте у випадку систем МН ситуація значно складніша, оскільки оператори мутації можуть передбачати необхідність перенавчання моделі, що є вкрай ресурсозатратним процесом. Для подолання цієї проблеми дослідники пропонують різні техніки оптимізації [5, с. 1160], такі як використання пакетної обробки, паралелізму GPU та аналізу на рівні сегментів. Однак навіть наве-

Таблиця 1

Порівняння різних рівнів мутацій для моделей МН

Рівень мутації	Приклад	Переваги	Недоліки
Рівень тренувальних даних	Додавання шуму в дані, видалення частини вибірки, маніпуляція розподілом класів	Може симулювати реальні помилки розробників	Не відповідає класичному визначенню мутаційного аналізу, бо перевіряє модель, а не тестову вибірку моделі
Рівень тренувальної програми	Зміна архітектури моделі, модифікація гіперпараметрів, зміна функції активації	Мутації мають прямий вплив на модель	Не відповідає класичному визначенню мутаційного аналізу та мутації на цьому рівні часто роблять модель непрацездатною
Рівень навченої моделі	Модифікація ваг моделі, видалення шарів, зміна вихідних значень нейронів	Найбільш наближена до поняття «вихідного коду» на етапі експлуатації	Зміни на цьому рівні важко співвіднести з типовими помилками розробників

дені оптимізації не завжди забезпечують бажану ефективність, а використання існуючих підходів до систем МН потребує емпіричної апробації.

Ще одним важливим викликом є оцінка мутацій. У моделях МН розмір тестових вибірок зазвичай значно перевищує обсяг тестів у класичних програмних системах. Як результат, велика кількість тестових зразків значно підвищує ймовірність «вбивства» мутантів, що може суттєво вплинути на точність оцінки ефективності тестового покриття. Деякі автори пропонують нові метрики оцінки тестового покриття. Частина з них засновані на кластеризації [3, с. 104], а інші є спеціалізованими показниками для різних типів моделей МН [5, с. 1160]. Ці оцінки мають на меті замінити мутаційну оцінку, однак, на думку автора доцільніше сфокусуватися на комбінованому підході, де різні види оцінок доповнюють одна одну, забезпечуючи комплексне охоплення перевірки якості тестового покриття.

Мутаційний аналіз є ітеративним процесом, спрямованим на поступове вдосконалення тестів та покращення їх здатності виявляти потенційні помилки. Однак у випадку систем МН, де тестування відбувається шляхом перевірки моделі на тестовому наборі, особливу складність становить оновлення тестів. Нові тестові дані мають не лише задовольняти вимоги системи, а й відображати реалістичні сценарії, які можуть виникнути в процесі експлуатації.

Таким чином, процес адаптації мутаційного аналізу до систем з МН має ряд викликів, зокрема, невідповідність класичним гіпотезам, розробка реалістичних мутаційних операторів, оптимізація швидкодії аналізу, створення більш точних метрик та еволюція тестового набору. Для подолання цих викликів необхідно використовувати комплексний підхід, що враховує особливості моделей МН, їхню стохастичну природу та різницю у їх життєвому циклі в порівнянні з традиційними системами.

Задля розв'язання поставлених проблем, пропонується список рекомендацій для дослідників і розробників програмного забезпечення, які допоможуть не лише інтегрувати мутаційний аналіз у процес тестування моделей МН, а й зберегти першооснови класичного методу:

Чітко проводити межі між вихідним кодом та тестовим набором під час проєктування операторів мутацій. Це дозволить уникнути проблемних ситуацій, коли мутаціям помилково піддається тестовий код, що може призвести до викривлення результатів аналізу.

Оцінювати реалістичність та ефективність мутаційних операторів шляхом виконання емпіричних досліджень. Тестування операторів на широкій вибірці допоможе виявити найбільш ефективні техніки.

Консультуватися з експертами у сфері розробки моделей МН. Співпраця з фахівцями може допомогти у розробці реалістичних помилок, визначенні поняття «маленьких змін» у контексті моделей МН, мінімізації ризиків неконтрольованих відхилень у поведінці моделі та впровадженні механізму оновлення тестових наборів.

Оцінювати мутаційний аналіз для моделей МН з використанням комбінованого підходу. Це дозволить враховувати як класичні метрики покриття коду, так і спеціалізовані показники для моделей МН, забезпечуючи повнішу оцінку ефективності тестового набору.

Висновки. У даному дослідженні розглянуто особливості використання мутаційного тестування до систем МН. Проведений аналіз продемонстрував, що підходи до тестування мутаціями, які використовуються у традиційних системах, потребують відповідної адаптації до систем МН. Також були визначені виклики, які виникають в процесі адаптації: визначення поняття «вихідного коду», ефективність й реалістичність мутаційних операторів, швидкодія тестування та неточність оцінки мутації. У ході дослідження були запропоновані рекомендації спрямовані на розв'язання вищезгаданих проблем.

Подальші дослідження у цій сфері можуть бути пов'язані з проєктуванням реалістичних мутаційних операторів, визначенням комплексного підходу до оцінки якості тестового набору, а також оптимізацією швидкодії та зниження ресурсозатратності тестування. Ще одним потенційним напрямом досліджень є розв'язання концептуальних проблем, які пов'язані із розбіжностями між класичними принципами мутаційного аналізу та специфікою систем МН.

Список літератури:

1. Jia Y., Harman M. An Analysis and Survey of the Development of Mutation Testing. IEEE Transactions on Software Engineering. 2011. Vol. 37, no. 5. P. 649–678. URL: https://doi.org/10.1109/tse.2010.62 (date of access: 23.02.2025).

2. Zhu Q., Panichella A., Zaidman A. A systematic literature review of how mutation testing supports quality assurance processes. Software Testing, Verification and Reliability. 2018. Vol. 28, no. 6. P. e1675. URL: https://doi.org/10.1002/stvr.1675 (date of access: 23.02.2025).

3. DeepMutation: Mutation Testing of Deep Learning Systems / L. Ma et al. 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), Memphis, TN, 15–18 October 2018. 2018. URL: https://doi.org/10.1109/issre.2018.00021 (date of access: 23.02.2025).

4. Shen W., Wan J., Chen Z. MuNN: Mutation Analysis of Neural Networks. 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Lisbon, 16–20 July 2018. 2018. URL: https://doi.org/10.1109/qrs-c.2018.00032 (date of access: 23.02.2025).

5. DeepMutation++: A Mutation Testing Framework for Deep Learning Systems / Q. Hu et al. 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA, 11–15 November 2019. 2019. URL: https://doi.org/10.1109/ase.2019.00126 (date of access: 23.02.2025).

6. Jahangirova G., Tonella P. An Empirical Evaluation of Mutation Operators for Deep Learning Systems. 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Porto, Portugal, 24–28 October 2020. 2020. URL: https://doi.org/10.1109/icst46399.2020.00018 (date of access: 23.02.2025).

7. Taxonomy of real faults in deep learning systems / N. Humbatova et al. ICSE '20: 42nd International Conference on Software Engineering, Seoul South Korea. New York, NY, USA, 2020. URL: https://doi.org/10.1145/3377811.3380395 (date of access: 23.02.2025).

8. Panichella A., Liem C. C. S. What Are We Really Testing in Mutation Testing for Machine Learning? A Critical Reflection. 2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), Madrid, ES, 25–28 May 2021. 2021. URL: https://doi.org/10.1109/icse-nier52604.2021.00022 (date of access: 23.02.2025).

9. DeMillo R. A., Lipton R. J., Sayward F. G. Hints on Test Data Selection: Help for the Practicing Programmer. Computer. 1978. Vol. 11, no. 4. P. 34–41. URL: https://doi.org/10.1109/c-m.1978.218136 (date of access: 23.02.2025).

10. Offutt A. J. Investigations of the software testing coupling effect. ACM Transactions on Software Engineering and Methodology (TOSEM). 1992. Vol. 1, no. 1. P. 5–20. URL: https://doi.org/10.1145/125489.125473 (date of access: 23.02.2025).

11. Hamlet R. G. Testing Programs with the Aid of a Compiler. IEEE Transactions on Software Engineering. 1977. SE-3, no. 4. P. 279–290. URL: https://doi.org/10.1109/tse.1977.231145 (date of access: 24.02.2025).

Lebediev A. V., Fedorova N. V. MUTATION ANALYSIS AND PROBLEMS OF ITS USE FOR MACHINE LEARNING MODELS

The article discusses the peculiarities of mutation analysis in machine learning (ML) systems. The analysis shows that classical mutation testing cannot be directly applied to ML systems without appropriate adaptation. The article describes in detail the main problems that arise during mutation analysis in ML systems, including the definition of 'source code', the development of efficient and realistic mutation operators, optimisation of the testing process speed and the need to improve methods for estimating mutation coverage. In addition, the article reviews the latest research on mutation analysis in the context of ML systems and provides a critical analysis of it.

The study offers a number of recommendations for researchers and software developers that will allow not only to integrate mutation testing into the ML models, but also to preserve the fundamental concepts of mutation analysis. In particular, the necessity of a clear distinction between source code and test suites, the creation of specialised mutation operators and the development of combined metrics for evaluating test coverage is substantiated. Moreover, the ways of optimising the computational complexity of mutation analysis used in existing tools are considered. The proposed solutions will help to improve the quality of testing of ML models and provide a more accurate assessment of their reliability.

The results of the study emphasise the importance of adapting classical testing methods to the features of ML systems to increase their efficiency and reliability. The conclusions determine the directions for further research in this area, including the development of new realistic mutation operators, optimisation of testing performance, and alignment of the main conceptual aspects of mutation analysis with the principles of operation of ML models. Further development of mutation analysis methods will make testing of ML models more reliable, structured and adaptive to the changing conditions of real-world applications

Key words: *mutation testing, mutation operators, quality of test sets, machine learning, machine learning models, processing of big data arrays.*