УДК 004.42
DOI https://doi.org/10.32782/2663-5941/2024.4/12

Дроменко В.Б.
Таврійський національний університет імені В.І. Вернадського

Лісовець С.М.
Таврійський національний університет імені В.І. Вернадського

ОСОБЛИВОСТІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРИ РОЗРОБЛЕННІ СИСТЕМ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

У даній роботі показано, що розробка систем штучного інтелекту є розширенням програмної інженерії та включає нові процеси й технології, які необхідні для розроблення та розвитку систем штучного інтелекту. Проаналізовано загальні проблеми, які пов'язані з розробленням застосувань штучного інтелекту та машинного навчання з точки зору інженерії програмного забезпечення. Виокремлені та наведені відмінності у процесах інженерії традиційного програмного забезпечення та програмного забезпечення з елементами штучного інтелекту та машинного навчання. Проаналізовано фактори, через які виникають певні проблеми та ризики для розробників програмного забезпечення і які слід брати до уваги при програмній реалізації моделей машинного навчання, щоб зробити цей процес передбачуваним та керованим.

Продемонстрована потреба у гармонізації дисциплін програмної інженерії та штучного інтелекту і машинного навчання з метою інтеграції життєвого циклу моделі штучного інтелекту та машинного навчання у процес розроблення програмного забезпечення.

Для систематизації загальних проблем та практик їх вирішення при розробці програмних систем з елементами штучного інтелекту та машинного навчання проаналізовано характеристики та запропоновані рекомендації для різних етапів життєвого циклу розроблення програмного забезпечення, а також рекомендації стосовно різних аспектів розроблення моделей штучного інтелекту та машинного навчання.

Запропоновані рекомендації допоможуть розробникам підвищити якість та ефективність розроблення моделей штучного інтелекту та машинного навчання, зменшити ризики та забезпечити узгодженість з найкращими практиками.

Ключові слова: *інженерія програмного забезпечення, штучний інтелект, машинне навчання, етапи життєвого циклу.*

Постановка проблеми. Програмні системи, які використовують елементи штучного інтелекту та машинного навчання, мають свої особливості порівняно з традиційними програмними системами. Через це виникають деякі проблеми та ризики для розробників програмного забезпечення.

Використання штучного інтелекту та машинного навчання для виконання системних функцій відрізняється від їх включення у вихідний код. Прогнозна здатність вбудовується у навчальні дані та використовується обмеженою кількістю викликів функцій. Сучасні методи контрольованого навчання, зокрема, використовують зворотне розповсюдження для налаштування вагових параметрів, що представляють розподіл інформації у глибоких нейронних мережах, з метою створення моделей штучного інтелекту та машинного навчання. Однак, на відміну від вихідного коду, який може бути розглянутий людиною, моделі штучного інтелекту та машинного навчання є складними конструкціями, які важко інтерпретувати. Це призводить до того, що деякі стандартні практики, такі як перегляд вихідного коду та вичерпне тестування, стають неефективними для моделей машинного навчання.

Аналіз останніх досліджень і публікацій. Стандарт ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) визначає програмну інженерію як «систематичний та дисциплінований підхід, який можна кількісно виміряти, до розробки, експлуатації та підтримки програмного забезпечення» [1].

Розроблення традиційного програмного забезпечення передбачає створення програмних кодів інженерами у вигляді вихідного коду, який можна розділити на функціональні одиниці, такі як класи, методи, функції тощо.

Системна та програмна інженерія досягли високого рівня зрілості, володіючи значним обсягом знань та практики, необхідної для ефективної розробки. Проте, ключовим викликом для обох галузей є їх адаптація до нових викликів цифрового середовища. Це дозволить створювати сучасні, безпечні, захищені, економічно ефективні та персоналізовані програмні продукти та послуги та зменшити технічну заборгованість таких систем.

Системи, що базуються на штучному інтелекті (англ. *artificial intelligence,* AI), представляють собою програмні системи, які мають принаймні один компонент AI, такий як системи розуміння природної мови, розпізнавання зображень або автономного водіння.

Розробка систем AI є розширенням програмної інженерії та включає нові процеси й технології, що є необхідними для розроблення та розвитку систем AI. На сьогоднішній день більшість систем AI використовують компоненти машинного навчання та не включають у свій склад експертні системи на основі правил, що відображає новий напрямок розвитку AI, зорієнтований на навчання на основі даних.

Машинне навчання (англ. *machine learning,* ML) – це набір статистичних технологій, які дають можливість комп'ютерам працювати з даними без явного програмування [2]. Технології ML поділяються на три основні типи:

– контрольоване навчання (*supervised learning*), де знаходиться відповідність між вхідними та вихідними даними (*input-output*) на основі розмічених даних. Такий тип AI/ML на теперішній час є домінуючим в промислових застосуваннях, хоча вимагає високоякісних даних;

– неконтрольоване навчання (*unsupervised learning*), що визначає шаблони в нерозмічених даних. Типовим застосуванням такого типу AI/ML є кластеризація;

– навчання з підкріпленням (*reinforcement learning*), що базується на використанні системи винагороди для оцінки продуктивності ML. Традиційно такий тип AI/ML застосовується у відеоіграх, але також набув поширення у застосуваннях для тестування програмного забезпечення та побудови самоадаптивних систем.

Наукові дослідження в галузі штучного інтелекту включають інженерію знань, автоматичне мислення, планування, оптимізацію, обробку природної мови, комп'ютерний зір, видобування шаблонів та інші напрямки. Ці дослідження мають значний вплив на класичні галузі, такі як охорона здоров'я, автомобілебудування, робототехніка, інформатика, відеоігри та фінанси, допомагаючи у розробці рішень для експертів та автоматизації промислових процесів [3, 4].

З іншого боку, методи контрольованого та неконтрольованого навчання є основними напрямками алгоритмів у штучному інтелекті та машинному навчанні для вирішення різних завдань, таких як класифікація, кластеризація, регресія, прогнозування або зменшення розмірності даних. Однак, досвід показує, що штучний інтелект та машинне навчання можуть бути успішно застосовані і в інших областях, таких як виявлення шаблонів, інтелектуальний аналіз даних, виявлення шахрайства або формування рекомендацій. Після встановлення наукових основ та технологічної підтримки для рішення завдань штучного інтелекту та машинного навчання через багато варіантів застосування в різних галузях, увага наукової спільноти зосереджується на визначенні нових обчислювальних моделей, таких як розподілений штучний інтелект, навчання з підкріпленням, метанавчання або багатоагентні системи.

Метою статті є аналіз загальних проблем, пов'язаних з розробленням застосувань штучного інтелекту та машинного навчання з точки зору інженерії програмного забезпечення. Оскільки розробка таких застосувань вимагає чітко визначеного процесу, розглянемо виклики та запропонуємо рекомендації для різних етапів життєвого циклу розроблення програмного забезпечення. Також проаналізуємо характеристики та розробимо рекомендації стосовно різних аспектів розроблення моделей штучного інтелекту та машинного навчання.

Виклад основного матеріалу дослідження. Життєвий цикл моделі штучного інтелекту та машинного навчання можна розглядати як процес, у якому потрібно працювати з даними, вибирати відповідну модель залежно від типу задачі та наявних даних, навчати та тестувати модель з різними конфігураціями та показниками продуктивності, а потім керувати навченою моделлю. Хоча цей процес може здаватися достатньо простим, на практиці виявляється, що це не так. Його складність полягає у потребі автоматизувати дії, які зазвичай виконуються вручну. Як і при розробці традиційних програмних систем, застосування інженерних принципів спрямоване на досягнення двох основних

покращень: забезпечення здатності до розвитку та підтримання життєвого циклу моделі штучного інтелекту та машинного навчання. Це особливо важливо, оскільки оцінка витрат і зусиль при розробці функцій штучного інтелекту та машинного навчання є неформалізованою.

Кожна функція штучного інтелекту та машинного навчання концептуально визначається, реалізується, перевіряється та працює протягом свого життєвого циклу, який складається з кількох етапів. Функція штучного інтелекту та машинного навчання може розглядатися як «чорна скринька», яка надає зовнішній інтерфейс операцій, які реалізовані через взаємодію між чотирма процесами, що складають життєвий цикл моделі штучного інтелекту та машинного навчання: керування даними, створення моделі, навчання та тестування, експлуатація (див. рис. 1).

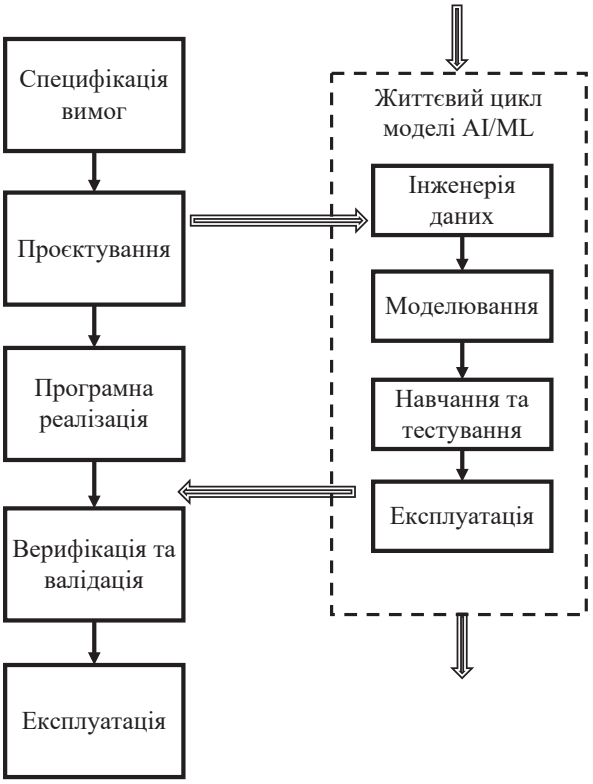

Рис. 1. Структурна схема загального процесу інженерії програмного забезпечення та життєвого циклу моделі штучного інтелекту та машинного навчання

З точки зору системної інженерії, функція штучного інтелекту та машинного навчання та її життєвий цикл представляють собою тип прозорої функціональності. Першим кроком у включенні функції штучного інтелекту та машинного навчання у процес розроблення систем є відкриття «чорної скриньки» і чітке представлення всіх процесів життєвого циклу штучного інтелекту та машинного навчання та їх взаємозв'язків для різних процесів технічної розробки. Для цього кожен процес повинен бути абстрактно описаним та стандартизованим, щоб концептуально описувати як дані, так і операції.

Операціоналізація та стандартизація практики розроблення програмного забезпечення є необхідними для економічно ефективного розвитку високоякісних та надійних систем штучного інтелекту та машинного навчання. Тому для досліджень та практичного застосування необхідно мати консолідований масив знань, що пов'язує проблеми та практики програмної інженерії, застосовані до розроблення систем штучного інтелекту та машинного навчання.

Програмні системи, переважно, специфікуються, проєктуються та впроваджуються у відповідності до дедуктивного та детерміністичного підходу, що означає різні подання (логічні, фізичні та процесні) для представлення та керування системою. Інтелектуальні системи, проте, можуть мати певну недетермінованість у своїй поведінці, оскільки деякі їхні частини керуються автоматичними процесами, базованими на даних, що врешті ускладнює окремі технічні процеси, такі як верифікація та валідація системи.

Таким чином, стає очевидною потреба у гармонізації дисциплін програмної інженерії та штучного інтелекту і машинного навчання з метою інтеграції життєвого циклу моделі штучного інтелекту та машинного навчання у процес розроблення програмного забезпечення. Отже, розробники програмного забезпечення повинні бути свідомі до різних проблем або ризиків, пов'язаних з програмними системами, що містять елементи штучного інтелекту та машинного навчання. Для систематизації загальних проблем та практик їх вирішення при розробці програмних систем з елементами штучного інтелекту та машинного навчання, розглянемо процес розроблення з двох точок зору:

1) розроблення програмного забезпечення;
2) розроблення моделі штучного інтелекту та машинного навчання.

Відмінності у процесах інженерії звичайного програмного забезпечення та програмного забезпечення з елементами штучного інтелекту та машинного навчання демонструє рис. 2.

Програмні системи, які містять компоненти штучного інтелекту та машинного навчання, потребують чітко визначеного процесу розро-

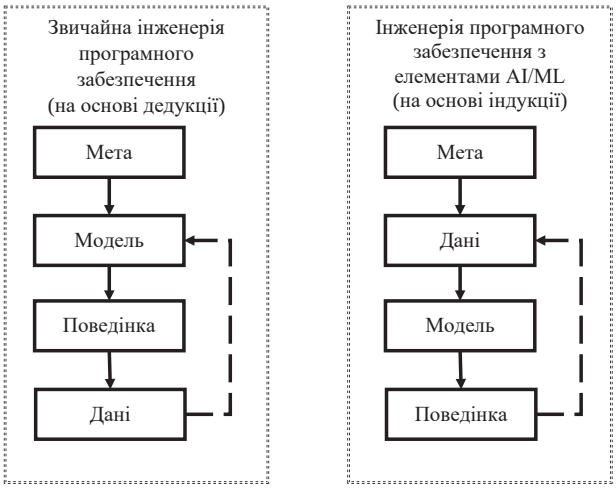

Рис. 2. Відмінності у процесах інженерії традиційного програмного забезпечення та програмного забезпечення з елементами штучного інтелекту та машинного навчання

блення та підтримки, але через унікальні характеристики систем штучного інтелекту та машинного навчання, етапи розроблення програмного забезпечення потребують адаптації для врахування конкретних вимог AI/ML.

1. **Розроблення вимог**. Крім традиційних дій з розроблення вимог, таких як аналіз, збір, специфікація та перевірка, розроблення вимог до систем штучного інтелекту та машинного навчання включає і специфічні вимоги. Низька якість вимог може призвести до численних проблем на подальших етапах [5]. Специфікація вимог до систем штучного інтелекту та машинного навчання є складним завданням в силу їхньої частої змінюваності. Тому, для мінімізації проблем етапу розроблення вимог, слід звернути увагу на наступні аспекти:

– аналіз доступних даних, оскільки важливо переконатися, що наявні дані придатні для забезпечення передбачуваних рішень на основі AI/ML. Тому, для ефективного формулювання вимог, крім фахівців із предметної області розроблюваної системи, потрібно залучати експертів з аналізу даних. Крім того, прототипування моделі штучного інтелекту та машинного навчання набуває особливої цінності, оскільки допомагає зрозуміти, які результати можна очікувати на основі доступних даних;

– формулювання вимог має бути чітким і враховувати як загальні вимоги до системи (наприклад, до швидкодії, забезпечуваної безпеки, надійності тощо), так і до результатів моделі AI/ML (наприклад, точність результатів, правдивість). Вимоги до систем AI/ML можуть часто змінюватися, тому розробники мають періодично переглядати та уточнювати їх;

– взаємодія з кінцевими користувачами є ключовим для адекватного визначення вимог, оскільки запізно виявлені пропущені вимоги можуть суттєво підвищити вартість розроблюваної системи;

– адаптація до зміни вимог з урахуванням того, що частота змін вимог до систем AI/ML може бути вищою, ніж до інших частин програмної системи, тому важливо регулярно переглядати та адаптувати їх.

2. **Проєктування** програмного забезпечення для систем з елементами штучного інтелекту та машинного навчання має свої особливості [6] через непередбачувану поведінку програм та велику кількість потрібних даних, на відміну від традиційних програмних систем, які мають кінцеву кількості станів і поведінка їх є передбачуваною. Розробка систем AI/ML має враховувати обмеження та накладні витрати на обробку даних. Алгоритми для машинного навчання розвиваються швидко темпами, тому дизайн програми AI/ML має бути гнучким, щоб прийняти ці зміни.

Попри відсутність змін у вимогах або відсутність помилок, з часом може погіршуватися продуктивність програм AI/ML. Отож, складно стає передбачити потреби в обслуговуванні програм AI/ML. Це означає, що проєкт має бути достатньо гнучким, щоб відповідати частим змінам. Наприклад, у випадку додавання можливостей AI/ML до наявної програми проєкт має передбачати мінімальні зміни наявної архітектури системи. Крім того, проєкт нової програмної системи з елементами AI/ML має бути гнучким для адаптації майбутніх змін.

На етапі проєктування систем AI/ML важливо врахувати наступні рекомендації:

– використання модульних архітектурних стилів завдяки поділу системи на модулі дозволяє краще розподілити завдання та забезпечити можливість повторного використання коду. Загальна система будується інтеграцією взаємодійних компонентів, які є набором функцій для вирішення чітко визначеної задачі;

– гнучкість проєкту через швидку зміну вимог та даних систем з елементами машинного навчання. Тому, проєкт має бути достатньо гнучким, щоб вносити зміни з мінімізацією зусиль та витрат;

– слабка залежність конструкції компонентів надає можливість мінімізувати потенційні витрати на супроводження. Це дозволяє розвивати компоненти незалежно один від одного;

– обробка великих обсягів даних має бути передбачена проєктом з відповідними механізмами у системі;

– стратегії інтеграції повинні бути чітко визначені у проєкті або в проєктній документації, оскільки моделі або компоненти машинного навчання мають бути інтегровані через відповідні інтерфейси.

Ці рекомендації допоможуть забезпечити ефективну розробку та підтримку програмних систем з елементами штучного інтелекту та машинного навчання.

3. На етапі **реалізації** програмного забезпечення з елементами штучного інтелекту та машинного навчання виникають складнощі із забезпеченням сумісності та цілісності системи [6], враховуючи різноманітність фреймворків і бібліотек для реалізації програм з елементами AI/ML. Такі моделі є «чорною скринькою», тому важко пояснити, а, відповідно, важко чітко усвідомити, чому вони працюють або не працюють.

Реалізація програмних систем AI/ML має враховувати вимоги цільової платформи при виборі фреймворків і бібліотек, оскільки середовище розроблення для моделей AI/ML може відрізнятися від робочого середовища. Слід також враховувати, що вибір реалізації має забезпечувати максимальну портативність, сумісність і адаптивність програм штучного інтелекту та машинного навчання з меншою вартістю.

Таким чином, для етапу реалізації є очевидними наступні рекомендації:

– спрямованість на використання цілісного набору фреймворків і бібліотек;

– врахування цільової платформи щодо сумісності, масштабованості та портативності програми.

4. Етап **інтеграції** має забезпечити функціональну цілісність системи шляхом об’єднання компонентів системи до єдиної узгодженої системи із бажаними функціями [7]. Інтеграція програм AI/ML може відбуватись за принципом двоетапного процесу: спочатку інтеграція компонентів підсистеми AI/ML, а потім інтеграція підсистем AI/ML з іншими підсистемами цільової системи. Тож, на процес і складність фази інтеграції може впливати визначений інтерфейс між компонентами AI/ML та іншими компонентами. Враховуючи, що моделі AI/ML і надалі постійно розвиватимуться, робочий процес AI/ML повинен сприяти безперервній інтеграції моделей AI/ML.

Тому на етапі інтеграції доречно врахувати такі рекомендації:

– для забезпечення функціональної цілісності системи інтеграція повинна об'єднувати компоненти системи таким чином, щоб забезпечити функціональну цілісність системи;

– врахування постійного розвитку моделей штучного інтелекту та машинного навчання і сприяння робочого процесу безперервній інтеграції моделей.

5. Етап **тестування**. Оскільки результати моделей машинного навчання мають стохастичний характер, виникають проблеми з отриманням детермінованих результатів для порівняння та перевірки програм машинного навчання [8]. З цієї причини наявні фреймворки модульного тестування не можуть бути використані для програм AI/ML. Крім того, моделі AI/ML навчаються на вхідних даних адаптивним та ітеративним способом і залежать від багатьох параметрів, таких як обрані функції, архітектура моделі та, навіть, дані навчання. Правила, які створені системами AI/ML, можуть бути невідомі розробникам. Все це значно ускладнює виявлення неправильної поведінки системи та точне визначення джерела помилок.

Необхідно також враховувати, що алгоритми AI/ML іноді можуть бути стійкими до певних помилок і надавати прийнятні результати шляхом помилок реалізації або компенсації «зашумлених» даних. Це також ускладнює виявлення та виправлення помилок в системі AI/ML. Тестування програмних систем AI/ML може потребувати великої кількості навчальних даних, а маркування таких даних вручну є витратним заходом. Крім того, випадковий вибір підмножин даних, ймовірно, не зможе визначити багато граничних випадків. Вищезазначені проблеми роблять тестування та виправлення неправильної поведінки в системах AI/ML достатньо складною задачею.

На етапі тестування слід брати до уваги наступні рекомендації:

– моделі штучного інтелекту та машинного навчання є непрозорими з важкопояснюваною поведінкою, тому вони потребують ретельного тестування для всіх можливих сценаріїв використання та в широкому діапазоні параметрів;

– помилки в системах штучного інтелекту та машинного навчання досить важко виявити через стохастичний характер моделей, тому всі підсистеми мають бути ретельно протестовані на рівні модульного та інтеграційного тестування.

6. Етап **розгортання** запускає програмну систему до експлуатації. Як правило, цей етап оновлює або замінює наявну систему, де може вини-

кати потреба в додаванні нових функціональних модулів програм машинного навчання до наявної системи [9]. Під час розгортання системи AI/ML слід враховувати, що платформа та інфраструктура для експлуатації системи можуть мати відмінності від середовища, в якому була навчена та оцінена модель AI/ML. Через такі відмінності можуть виникнути проблеми сумісності, переносності та масштабованості, що значною мірою може вплинути на продуктивність системи.

Очевидні рекомендації, які слід брати до уваги на етапі розгортання:

− урахування відмінностей платформ між середовищами розроблення та експлуатації під час розгортання системи штучного інтелекту та машинного навчання;

− врахування вимог до переносності та масштабованості при розгортання систем штучного інтелекту та машинного навчання в експлуатаційному середовищі;

− обережне розгортання систем штучного інтелекту та машинного навчання для мінімізації впливу на користувачів.

Відмінності у характеристиках та вимогах програмних систем AI/ML потребують чітко визначеного набору принципів і рекомендацій. Слід пам’ятати, що алгоритми машинного навчання пропонують рішення загального призначення. Лише вірне формулювання задачі дозволить визначити адекватний алгоритм, адже невірне формулювання задачі може призвести до принципової неможливості коректної роботи програмної системи AI/ML. Отже, правильна постановка задачі є запорукою успіху інших етапів розроблення систем AI/ML.

При *формулюванні задачі* слід спиратись на наступне:

− чітке розуміння очікуваних результатів та характеристик даних, на якому базується правильне формулювання задачі машинного навчання;

− достатнє розуміння розробниками алгоритмів машинного навчання для вибору найбільш підходящого алгоритму для конкретної задачі.

Збір даних є критичними накладними витратами для машинного навчання через їх великі обсяги даних. Але, недостатня кількість даних також є проблемою для програмних систем AI/ML. Збір і обробка великого обсягу даних мають бути зосереджені на представництві повного діапазону поведінки (повноті), правильності даних (точності), відсутності суперечливих даних (узгодженості) та відповідності поточному стану системи (своєчасності) даних для забезпечення їх якості. Однак, підтримка якості даних вимагає ретельного збору, контролю та обслуговування даних, що часто є дороговартісним з огляду на час і пов'язані з цим обчислювальні роботи.

Тому, слід враховувати наступне щодо *збору даних*:

− забезпечення повноти, точності, узгодженості і своєчасності даних в процесі збору даних;

− гнучкість процесу збору даних для адаптації до змін структури даних, яка може змінюватися з часом;

− аналіз вимог до даних перед початком збору може зекономити час і ресурси в подальшому процесі машинного навчання.

Необроблені дані потребують *попередньої обробки* з метою очищення, організації та трансформації (за потреби), оскільки можуть бути непридатними для використання в моделях машинного навчання. Головною проблемою для алгоритмів машинного навчання вважаються саме необроблені дані, оскільки вони можуть негативно впливати на процес навчання та отримані результати. Попередня обробка даних є важливим та трудомістким етапом для машинного навчання в силу того, що потребуватиме значних витрат часу та зусиль.

Отже, при *попередній обробці* даних слід врахувати такі рекомендації:

− очищення, заповнення відсутніх значень та інші необхідні перетворення забезпечують якість даних;

− максимізація повторного використання оброблених даних, оскільки попередня обробка може бути спільною для кількох елементів AI/ML.

Виділення ознак спрямоване на оптимальне перетворення вхідних даних у вектори ознак для алгоритмів машинного навчання. Цей етап виокремлює набір ознак, які найкраще представляють приховані характеристики даних для машинного навчання і спрямований на усунення потенційних шумів і надмірності даних. Отримання даних у «низькорозмірному» поданні збільшує швидкість навчання та дозволяє зробити результати роботи алгоритмів машинного навчання візуальними. Проте, некоректно обрані ознаки можуть суттєво погіршити отримувані за допомогою реалізованих моделей результати.

При *виділенні ознак* слід врахувати, що:

− якість набору ознак суттєво впливає на продуктивність моделей машинного навчання, тому розробники повинні виокремити найкращий набір ознак;

– створення автоматизованого конвеєра для виділення ознак дозволить автоматизувати процес виділення ознак та ефективно використовувати час та ресурси;

– ознаки потрібно очистити від забруднень, оскільки це може негативно вплинути на продуктивність моделі;

– мінімізація кількості ознак в наборі, який зберігає приховані в даних властивості, оскільки кількість виділених ознак пов'язана зі складністю моделей;

– періодичний перегляд наборів ознак для підтримки їх актуальності і ефективності.

Побудова моделі AI/ML відбувається на основі конкретних алгоритмів машинного навчання в залежності від поставленої задачі та характеристик даних шляхом використовування наявних моделей з бібліотек або створенням власних моделей. В подальшому моделі ітераційно навчаються, доки не досягнуть бажаного рівня продуктивності. Поширеною проблемою для моделей AI/ML є так зване «перенавчання», коли модель добре працює на навчальному наборі даних, але погано узагальнюється з іншими даними. Причиною такого може стати занадто складна модель. Рішенням подолання проблеми є знаходження найпростішої моделі, яка забезпечує потрібну продуктивність. Проте, скоріш за все, занадто прості моделі машинного навчання будуть недостатніми і зафіксувати приховані закономірності в даних не зможуть. Також слід враховувати і розподіл даних, який має бути збалансованим, інакше висновок моделі зміщуватиметься в бік класу, що є домінуючим в навчальних даних.

Таким чином, основними рекомендаціями при *побудові моделі* є:

– адаптація алгоритмів машинного навчання у відповідності до конкретної задачі та характеристик даних;

– градація складності моделі (починаючи з найпростіших рішень та поступовий перехід до складніших), збалансовуючи компроміс між ресурсами та продуктивністю;

– важливо забезпечити якість і збалансований розподіл даних, щоб уникнути зміщення висновків моделі в бік домінуючого класу;

– перед розробкою нових моделей розгляд можливості повторного використання наявних рішень для підвищення продуктивності.

Оцінка моделей машинного навчання відбувається шляхом їх застосування до верифікаційного набору даних, який є відокремленим від набору даних для навчання. Оскільки через зміни в характеристиках вхідних даних з часом продуктивність може знижуватись, важливим є не лише оцінювання перед розгортанням, а й безпосередньо моніторинг продуктивності після розгортання. Тобто, щоб адаптуватися до змін моделі машинного навчання можуть потребувати оновлення (наприклад, перенавчання) і тоді оцінка моделей машинного навчання може стати ітеративною протягом життєвого циклу. Крім того, в програмній системі AI/ML можуть бути застосовані різні взаємодіючі моделі. Отже, продуктивність окремих моделей може являти лише частину наскрізних сценаріїв і важливо отримати оцінки як на рівні окремих моделей, так і на рівні всієї системи.

При *оцінюванні моделей* слід дотримуватися наступних рекомендації:

– повний набір верифікаційних даних, щоб представляти різноманітні можливі сценарії використання;

– оцінка моделі повинна містити не лише точність, а й інші характеристики, такі як пропускна здатність, використання ресурсів і масштабованість;

– оцінка продуктивності як на рівні окремих моделей, так і на рівні всієї системи при взаємодії різних моделей.

Наступним кроком життєвого циклу для навчених моделей є їх *інтеграція* до цільової системи з метою об'єднання всіх необхідних компонентів (наприклад, моделей, конвеєрів введення-виведення). Застосування кількох моделей може потребувати визначення та реалізації інтерфейсів для кожної окремої моделі для забезпечення їхньої взаємодії з іншими моделями та компонентами. Поширеним підходом є розгортання моделей машинного навчання як сервісів (*service*) з доступом до сервісів через API. При розгортанні моделей ML слід враховувати портативність і сумісність моделей щодо цільової платформи.

Під час *інтеграції та розгортання* моделі доречними будуть такі рекомендації:

– при інтеграції компонентів машинного навчання важливо забезпечити функціональну цілісність програмних систем AI/ML;

– під час розгортання моделей слід враховувати сумісність і портативність платформ розроблення та експлуатації;

– розгортання моделей машинного навчання повинно забезпечити плавні зміни в наявній системі без впливу на користувача або бізнес-процеси.

Управління моделлю машинного навчання поєднує її навчання, підтримку, розгортання,

моніторинг і документування, що з точки зору робочого процесу є складним завданням. Моделі ML керуються даними та базуються на різноманітних припущеннях щодо розподілу та шаблонів даних. При цьому початкові характеристики даних можуть не зберігатися через зміни в даних, і це, з великою ймовірністю, вплине на поведінку моделі. Таким чином, важливо контролювати продуктивність розгорнутих моделей шляхом відстежування зміни в характеристиках даних, а також перенавчанням та повторної перевірки моделі. Для реалізації такого контролю потрібні ітерації життєвого циклу моделі машинного навчання, що вимагає великих витрат часу та ресурсів.

Також важливим є забезпечити відстеження версій моделі, наборів даних і конфігурації, що забезпечить відтворюваність моделей машинного навчання, а, відповідно, і полегшить керування робочим процесом. Відтворюваність моделі буде корисною для аналізу та порівнювання поведінки та продуктивності моделі, а також підтримки рішення щодо розгортання.

Слід дотримуватися таких рекомендацій щодо управління моделлю:

− моніторинг продуктивність розгорнутих моделей та забезпечення їхньої підтримку після розгортання;

− відстеження змін в характеристиках даних та розподілі даних для ефективного управління моделлю;

− забезпечення версіонування моделей, даних і конфігурацій, а також належне документування кожної фази життєвого циклу системи AI/ML для полегшення відтворюваності та підтримки моделей машинного навчання.

Етика в розробці AI/ML спрямована на зменшення негативних наслідків впровадження елементів штучного інтелекту та машинного навчання в різні сфери застосування програмного забезпечення. Дослідники та практики повинні відповідально використовувати AI, а при дослідженні та розробці програмної системи AI/ML, слід дотримуватися стандартного кодексу етики програмної інженерії [10].

Загалом, рекомендації щодо етики в розробці AI є наступними:

− відповідальне використання штучного інтелекту за принципами етики при розробленні програмних систем AI/ML;

− збереження конфіденційності та безпеки особистих і бізнесових даних під час використання AI/ML;

− пріоритет колективного добробуту над бізнес-вигодами при використанні AI/ML.

Висновки. В силу того, що останнім часом відслідковується значне зростання інтересу до програмних систем з елементами штучного інтелекту та їх застосування в різноманітних бізнесових сферах, врахування специфічних особливостей таких технологій у життєвому циклі розроблення програмного забезпечення є ключовим для забезпечення успішності та ефективності проєктів.

Життєвий цикл розроблення програмного забезпечення сьогодення є достатньо формалізованим завдяки визнаним добіркам рекомендацій та найкращих практик. Проте, при розробленні програмних систем з елементами AI/ML такі рекомендації потребують певних модифікацій.

Інженерія програмного забезпечення з елементами штучного інтелекту та машинного навчання потребує глибокого розуміння як основ програмування, так і алгоритмів машинного навчання. Такий підхід допомагає забезпечити не лише функціональність системи, але й її відповідність бізнес-потребам та етичним нормам.

З цією метою в роботі проаналізовано особливості розроблення систем з елементами AI/ML з точки зору інженерії програмного забезпечення, виокремлені відмінності у процесах інженерії традиційного програмного забезпечення та програмного забезпечення з елементами штучного інтелекту та машинного навчання. Крім того, проаналізовано фактори, які слід брати до уваги при програмній реалізації моделей машинного навчання, щоб зробити цей процес передбачуваним та керованим.

В подальшому докладні системні рекомендації до процесу розроблення програмного забезпечення з елементами AI/ML стануть практично корисними для команд, які працюють у цій сфері. Ці рекомендації допоможуть підвищити якість та ефективність розроблення, зменшити ризики та забезпечити узгодженість з найкращими практиками.

Список літератури:

1. Сайт SEVOCAB: Software and Systems Engineering Vocabulary [Електронний ресурс]. – Режим доступу: https://pascal.computer.org/sev_display/index.action

2. Харченко В. О. Основи машинного навчання : навч. посіб. / В. О. Харченко. – Суми : Сумський державний університет, 2023. 264 с.

3. Alvarez-Rodriguez J. M., Zuniga R. M., Pelayo V. M., Llorens, J. Challenges and opportunities in the integration of the Systems Engineering process and the AI/ML model lifecycle. In INCOSE International Symposium. 2019, Vol. 29, no. 1. P. 560-575.

4. Challa H., Niu N., Johnson R. Faulty Requirements Made Valuable: On the Role of Data Quality in Deep Learning. IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). 2020. P. 61-69.

5. Vogelsang A., Borg M. Requirements Engineering for Machine Learning: Perspectives from Data Scientists. IEEE 27th International Requirements Engineering Conference Workshops (REW). 2019. P. 245-251.

6. Washizaki H., Khomh F., Gueheneuc Y.-G., Takeuchi H., Natori N., Doi T., Okuda S. Software-Engineering Design Patterns for Machine Learning Applications. Computer. 2022. Vol. 55, no. 3. P. 30-39.

7. Martinez-Fernandez S., Bogner J., Franch X., Oriol M., Siebert J., Trendowicz A., Vollmer A. M., Wagner S. Software Engineering for AI-Based Systems: A Survey. ACM Transactions on Software Engineering and Methodology. 2022. Vol. 31, no. 2, article 37e.

8. Nishi Y., Masuda S., Ogawa H., Uetsuki K. A Test Architecture for Machine Learning Product. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). 2018. P. 273-278.

9. Paleyes A., Urma R.-G., Lawrence N. D. Challenges in Deploying Machine Learning: a Survey of Case Studies. ACM Computing Surveys. 2022, April. Available at: https://dl.acm.org/doi/10.1145/3533378 [Accessed: 26.04.2024].

10. Code of Ethics. IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices. 1999. Available from: https://www.computer.org/education/code-of-ethics [Accessed: 26.04.2024].

Dromenko V.B., Lisovets S.M. FEATURES OF SOFTWARE ENGINEERING IN THE DEVELOPMENT OF SYSTEMS WITH ELEMENTS OF ARTIFICIAL INTELLIGENCE

This work shows that the development of artificial intelligence systems is an extension of software engineering and includes new processes and technologies that are necessary for the development and development of artificial intelligence systems. The general problems associated with the development of artificial intelligence and machine learning applications from the point of view of software engineering are analyzed. The differences in the engineering processes of traditional software and software with elements of artificial intelligence and machine learning are highlighted and given. The factors that cause certain problems and risks for software developers and which should be taken into account in the software implementation of machine learning models in order to make this process predictable and manageable are analyzed.

The need to harmonize the disciplines of software engineering and artificial intelligence and machine learning in order to integrate the life cycle model of artificial intelligence and machine learning into the software development process is demonstrated.

In order to systematize common problems and practices for solving them in the development of software systems with elements of artificial intelligence and machine learning, the characteristics and proposed recommendations for various stages of the software development life cycle were analyzed, as well as recommendations regarding various aspects of the development of artificial intelligence and machine learning models.

The proposed guidelines will help developers improve the quality and efficiency of AI and machine learning model development, reduce risk, and ensure alignment with best practices.

Key words: *software engineering, artificial intelligence, machine learning, life cycle stages.*