УДК 004.42:004.89 https://doi.org/10.15407/pp2022.03-04.099

ПРО ДЕЯКІ АСПЕКТИ ІНЖЕНЕРІЇ ПРОГРАМНИХ СИСТЕМ З ЕЛЕМЕНТАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Віра Любченко

Програмні системи з елементами AI/ML мають деякі відмінні характеристики у порівнянні з традиційними програмними системами. Таким чином, для розробників програмного забезпечення з'являється кілька проблем і факторів ризику щодо розробки застосувань цього типу. В роботі проаналізовано загальні проблеми розробки AI/ML-застосувань з точок зору інженерії програмного забезпечення та машинного навчання. Оскільки AI/ML-застосування потребують чітко визначеного процесу розробки програмного забезпечення, розглянуто виклики та рекомендації для різних етапів життєвого циклу розробки програмного забезпечення. Оскільки розробка моделей AI/ML має певні аспекти, які слід розглядати як проєкт розробки програмного забезпечення, розглянуто характеристики та рекомендації щодо різних діяльностей для її розробки.

Ключові слова: програмна інженерія, штучний інтелект, машинне навчання, розробка вимог, проєктування, реалізація, інтеграція, тестування, розгортання.

AI-based software systems are rapidly spreading in various business areas. In this context, the unavoidable convergence of the Software Engineering and Artificial Intelligence and Machine Learning (AI/ML) disciplines is considered an obvious and one of the following significant challenges within the engineering process. The life cycle, models, and technologies of AI/ML elements are pretty specific, and this should be considered in software engineering to ensure their performance and compliance with business needs. AI/ML applications have some distinct characteristics compared to traditional software applications. Thus, several challenges and risk factors regarding AI/ML applications appear to software developers. To study the common challenges in AI/ML application development, we used two different perspectives: software engineering and machine learning. AI/ML applications, like other software systems, need a well-defined software engineering process for their development and maintenance. We discussed challenges and recommendations for different phases of the software development life cycle for ML applications, particularly requirement engineering, design, implementation, integration, testing, and deployment. AI/ML application development has specific aspects to consider as a software development project. We discussed the characteristics and recommendations concerning problem formulation, data acquisition, preprocessing, feature extraction, model building, evaluation, model integration and deployment, model management, and ethics in AI/ML development. In the work, there were formulated recommendations for each analyzed challenge that should be useful for software developers. The next stage of this research is the compilation of detailed systematic guidelines for the software development process for AI/ML systems.

Keywords: software engineering, artificial intelligence, machine learning, requirement engineering, design, implementation, integration, testing, deployment.

Вступ

ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) визначає програмну інженерію як «застосування систематичного, дисциплінованого підходу, що піддається кількісному вимірюванню, до розробки, експлуатації та підтримки програмного забезпечення».

Розробка звичайного програмного забезпечення полягає в реалізації програм інженерами у формі вихідного коду, який можна розкласти на функціональні одиниці, як-от, класи, методи, функції тощо.

Дисципліни системної та програмної інженерії досягли високого рівня зрілості. Існує величезний масив знань і практики, які забезпечують наукові та технологічні основи для правильного процесу розробки. Однак одним із головних наріжних каменів обох дисциплін є їх адаптація до викликів, пов'язаних із новим цифровим середовищем, що розвивається. Така адаптація дозволить надавати сучасні, безпечні, захищені, економічно ефективні та персоналізовані продукти та послуги на основі програмного забезпечення та долати «прихований» технічний борг таких систем.

Системи на основі штучного інтелекту (artificial intelligence – AI) – це програмні системи з функціями, що підтримуються принаймні одним компонентом AI (наприклад, системи розпізнавання зображень, розуміння природної мови, автономного водіння).

Галузь розробки систем AI є розширенням програмної інженерії та включає нові процеси й технології, необхідні для розробки та еволюції систем AI. Наразі більшість систем AI реалізують компоненти машинного навчання і не містять експертні системи на основі правил. Це відповідає новій хвилі AI, яка пов'язана з навчанням на основі даних.

Машинне навчання (machine learning – AI/ML) – це набір статистичних технологій для забезпечення комп'ютерів здатністю працювати з даними без явного програмування [1]. Технології AI/ML заведено розділяти на три основні типи. Контрольоване навчання (supervised learning) знаходить відповідність між парами вхід-вихід (input-output) на основі «розмічених» даних із правильною відповіддю. Цей тип AI/ML зараз домінує в промислових застосуваннях, але вимагає високоякісних даних. Неконтрольоване навчання (unsupervised learning) визначає шаблони в «нерозмічених» даних. Типовим застосуванням цього типу AI/ML є кластери-

зація. Навчання з підкріпленням (reinforcement learning) ґрунтується на застосуванні функції винагороди для кількісної оцінки продуктивності AI/ML за принципом батога й пряника. Основною ділянкою застосування цього типу AI/ML традиційно є відеоігри, але існують і застосування, наприклад, для тестування програмного забезпечення та побудови самоадаптивних систем.

Використання AI/ML для забезпечення системних функцій принципово відрізняється від їх інкапсуляції у вихідний код. Прогнозна здатність вбудована в навчальні дані та використовується за допомогою обмеженої кількості викликів функцій. Сучасне контрольоване навчання використовує зворотне розповсюдження, щоб підібрати вагові параметри, які представляють поширення інформації в глибоких нейронних мережах, як результат чого створюється модель AI/ML. У порівнянні з вихідним кодом, який читається людиною, моделі AI/ML є непрозорими конструкціями, які важко інтерпретувати. Відповідно, деякі стандартні практики, такі як перегляд вихідного коду та вичерпне тестування, втрачають сенс застосовування до моделей машинного навчання.

Програмна інженерія та штучний інтелект

З одного боку, такі галузі AI, як автоматичне мислення, інженерія знань, планування, оптимізація, обробка природної мови, видобування шаблонів, комп'ютерний зір, були одними з наукових сфер, в яких досліджувалося застосування різних типів алгоритмів AI, таких як генетичне програмування, штучні нейронні мережі, оптимізація мурашиних колоній тощо. Такі сфери, як охорона здоров'я, інформатика, автомобілебудування, робототехніка, відеоігри чи фінанси, були й залишаються класичними сферами, на які AI/ML мали значний вплив завдяки розробці рішень, які допомагали експертам у процесах прийняття рішень або дозволяли автоматизувати промислову діяльність [2].

З іншого боку, методи контрольованого та неконтрольованого навчання були основними категоріями алгоритмів AI/ML для розв'язання проблем, пов'язаних із кластеризацією, класифікацією, регресією, прогнозуванням або зменшенням розмірності. Проте досвід показує, що AI/ML може успішно застосовуватися для розв'язання задач інтелектуального аналізу даних, видобування шаблонів, виявлення шахрайства або побудови рекомендацій. Після того, як наукові основи та технологічна підтримка розв'язання задач AI/ML були створені та перевірені за допомогою багатьох варіантів використання в багатьох областях, увага наукової спільноти сфокусувалася на визначенні нових обчислювальних моделей, таких як розподілений штучний інтелект, навчання з підкріпленням, метанавчальні або багатоагентні системи.

У цьому контексті життєвий цикл моделі AI/ML можна розглядати як процес, у якому необхідно мати справу з даними, вибирати цільову модель залежно від типу задачі та доступних даних, навчати та тестувати модель відповідно до різних конфігурацій та показників продуктивності та, нарешті, керувати навченою моделлю. Хоча процес не виглядає особливо складним, практика показала, що це не так просто. Складність пов'язана з автоматизацією дій, які зазвичай виконуються вручну. Як і при розробці традиційних програмних системах, застосування інженерних принципів має допомогти досягти двох основних удосконалень: забезпечити здатність розвиватися та підтримувати життєвий цикл моделі AI/ML. Це особливо актуально, оскільки оцінка витрат і зусиль при розробці функції AI/ML не формалізована.

Концептуально будь-яка функція AI/ML визначається, реалізується, перевіряється та працює протягом життєвого циклу, який складається з кількох етапів. Функція AI/ML – це чорна скринька, яка забезпечує зовнішній інтерфейс операцій, реалізованих через взаємодію між чотирма процесами, що складають життєвий цикл моделі AI/ML: керування даними, створення моделі, навчання та тестування та експлуатація (рис. 1).

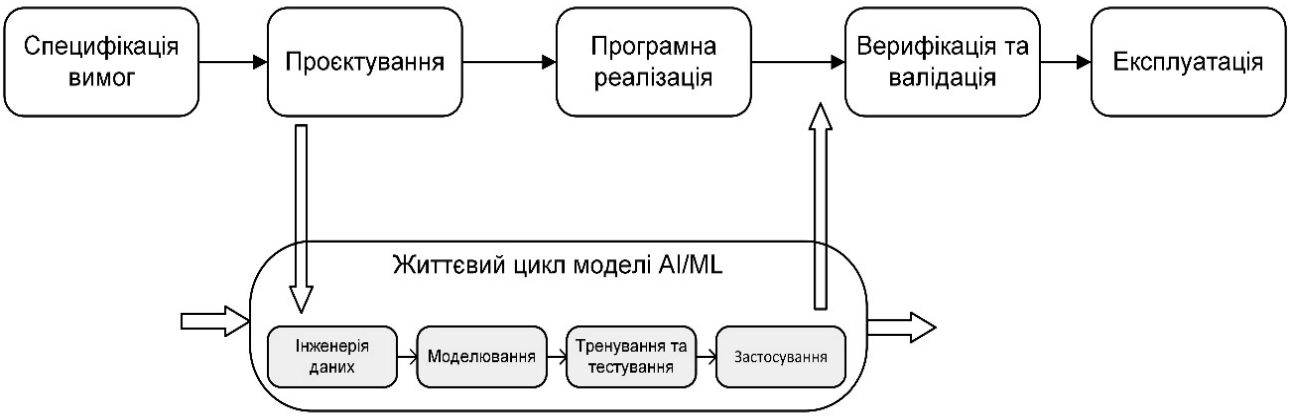

Рис. 1. Загальний процес інженерії програмного забезпечення та життєвий цикл моделі AI/ML

З точки зору процесу системної інженерії, функція AI/ML і її життєвий цикл є типом прозорої функціональності. Перший крок до того, щоб зробити функцію AI/ML частиною процесу розробки систем, – це відкрити чорну скриньку і зробити всі процеси життєвого циклу AI/ML та його залежності явно доступними

для різних процесів технічної розробки. Для цього кожен процес має бути описаний абстрактно та стандартизовано, що дозволяє концептуально описувати як дані, так і операції.

Операціоналізація та стандартизація практики розробки програмного забезпечення необхідні для економічно ефективного розвитку високоякісних і надійних систем AI/ML. Отже, як для досліджень, так і для практичного застосування, необхідно мати консолідований масив знань, який пов'язує проблеми та практики програмної інженерії, застосовані для розробки систем AI/ML.

Зазвичай програмні системи специфікуються, проєктуються та впроваджуються відповідно до дедуктивного та детерміністичного підходу, що означає різні подання (логічні, фізичні та процесні) для представлення та керування системою. Інтелектуальні системи приносять певну недетермінованість поведінки. Деякі частини системи керуються автоматичними процесами, базованими на даних, що ускладнює окремі технічні процеси, такі як верифікація та валідація системи.

Отже, виникає необхідність гармонізувати дисципліни програмної інженерії та AI/ML, щоб інтегрувати життєвий цикл моделі AI/ML у процес розробки програмного забезпечення. Тому розробники програмного забезпечення мають усвідомлювати низку проблем або факторів ризику щодо програмних систем з елементами AI/ML. Щоб систематизувати загальні проблеми та практики їх вирішення при розробці програмних систем з елементами AI/ML, розглянемо процес розробки з двох точок зору – розробки програмного забезпечення та розробки моделі AI/ML.

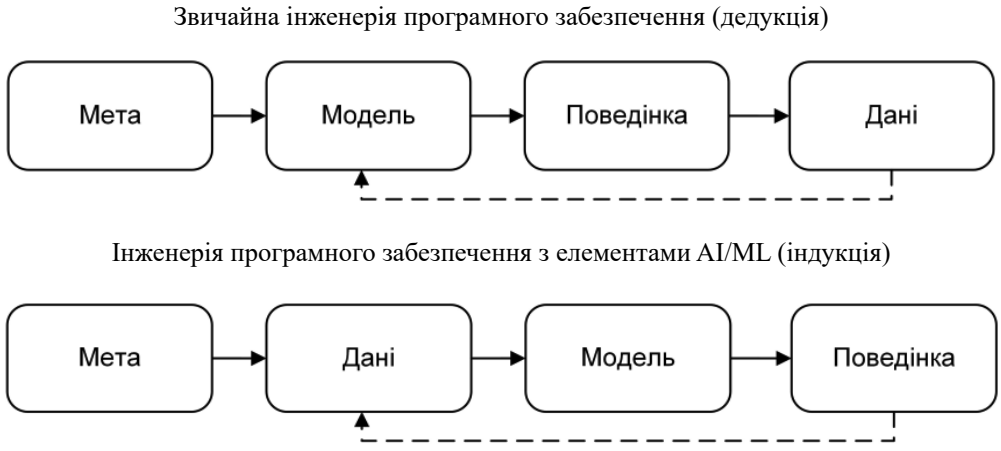

Рис. 2. Різниця в процесах інженерії звичайного програмного забезпечення та програмного забезпечення з елементами AI/ML

Проблеми та рекомендації інженерії програмного забезпечення

Програмні системи з елементами AI/ML, як і будь-які інші програмні системи, потребують чітко визначеного процесу розробки та підтримки. Однак, враховуючи відмінні характеристики систем AI/ML, фази процесу розробки програмного забезпечення потребують коригування для відповідності конкретним вимогам AI/ML.

Фаза 1: розробка вимог [3]. Низька якість вимог може призвести до багатьох проблем на наступних етапах розробки програмного забезпечення. Розробка вимог для систем AI/ML включає як специфічні, так і традиційні дії з розробки вимог, такі як аналіз здійсненності, збір вимог, специфікація вимог і перевірка. Оскільки вимоги до систем AI/ML можуть часто змінюватися, специфікація вимог до них є складним завданням.

Щодо розв'язання проблем фази розробки вимог, слід звернути увагу на такі аспекти.

– Моделі AI/ML керуються даними, тому важливо проаналізувати, чи придатні наявні дані для передбачуваних рішень на основі AI/ML. Тому надзвичайно важливо залучати до процесу розробки вимог не лише фахівців із предметної області розроблюваної системи, а і фахівців з аналізу даних. Також цінності набуває прототипіювання моделі AI/ML, яке уможливлює розуміння того, які результати принципово можуть бути отримані на основі доступних даних.

– Вимоги формулюються як до програмної системи в цілому (наприклад, до забезпечуваної надійності, безпеки, швидкості роботи тощо), так і до результатів роботи моделі AI/ML (наприклад, правдивість і точність результатів). Програми можуть ставити вимоги як до специфічних, так і до інших типів. Розробники повинні знати про будь-які суперечливі вимоги та відповідно їх адаптувати.

– Безпосередній зв'язок із кінцевими користувачами надзвичайно важливий для адекватного визначення вимог. Пізнє виявлення пропущених вимог може суттєво збільшити вартість розроблюваної системи.

– Вимоги до програмних систем AI/ML можуть часто змінюватися. Відповідно набуває особливої важливості періодичне уточнення вимог. Тут слід взяти до уваги те, що частота зміни вимог до підсистеми AI/ML зазвичай більша, ніж частота зміни вимог до решти програмної системи.

Фаза 2: проєктування [4]. Проєкт програмного забезпечення визначає деталі масштабу (scope), функціональних можливостей і взаємодії компонентів програмного забезпечення. Традиційні програмні системи

мають кінцеву кількості станів, і поведінка системи є передбачуваною. Однак для систем AI/ML поведінка програми є непередбачуваною та визначається навчальними даними. Це робить розробку систем AI/ML складним завданням. Також системи AI/ML потребують великої кількості даних. Таким чином, розробка систем AI/ML повинна враховувати обмеження та накладні витрати на обробку даних. Оскільки алгоритми та фреймворки для машинного навчання швидко розвиваються, дизайн програми AI/ML має бути гнучким, щоб прийняти ці зміни.

До того ж програми AI/ML керуються даними, і їхня продуктивність може погіршуватися з часом, попри відсутність змін у вимогах або відсутність помилок.Отож, може бути важко передбачити потреби в обслуговуванні програм AI/ML. Тобто проєкт має бути настільки гнучким, щоб відповідати частим змінам. У випадках додавання можливостей AI/ML до наявної програми проєкт повинен передбачати мінімальні зміни наявної архітектури системи. Також проєкт нової програмної системи з елементами AI/ML має бути гнучким, щоб адаптувати майбутні зміни.

На фазі проєктування слід користатися такими рекомендаціями.

– Для проєктування систем AI/ML слід застосовувати модульні архітектурні стилі, оскільки модульність забезпечує поділ проблем і можливість повторного використання. Тобто система поділяється на компоненти, які об'єднують функції для вирішення чітко визначеної задачі. Загальна система будується шляхом інтеграції взаємодійних компонентів.

– Компоненти AI/ML можуть розвиватися швидше, ніж інші компоненти системи, через часті зміни вимог і даних. Тобто проєкт повинен бути достатньо гнучким для того, щоб вносити зміни з мінімальними зусиллями та витратами. Конструкція компонентів AI/ML має бути слабо пов'язана, щоб знизити потенційні витрати на супроводження.

– Програми AI/ML потребують великих обсягів даних. Таким чином, проєкт повинен передбачати відповідний механізм обробки даних у системі.

– Оскільки моделі або компоненти мають бути інтегровані через відповідні інтерфейси, стратегії інтеграції мають бути показані в проєкті або в проєктній документації.

Фаза 3: реалізація [5]. Розробка програм AI/ML зазвичай передбачає використання багатьох різних фреймворків і бібліотек. Складність полягає в необхідності зібрати різноманітний набір фреймворків і бібліотек і забезпечити сумісність і цілісність системи. Моделі AI/ML є «чорною скринькою» і їх важко пояснити. Відповідно важко чітко зрозуміти, чому вони працюють або не працюють.

Також середовище розробки для моделей AI/ML може відрізнятися від робочого середовища. Таким чином, реалізація програмних систем AI/ML повинна враховувати вимоги цільової платформи при виборі фреймворків і бібліотек. Слід також враховувати, що апаратно-програмна екосистема для програм машинного навчання швидко розвивається. Тобто вибір реалізації має забезпечувати максимальну портативність, сумісність і адаптивність програм AI/ML з меншою вартістю.

Отже, для фази реалізації рекомендації очевидні:

– реалізація програм AI/ML має бути спрямована на використання цілісного набору фреймворків і бібліотек;

– реалізація повинна враховувати цільову платформу програми щодо масштабованості, сумісності та портативності.

Фаза 4: інтеграція [5]. Процес інтеграції програмного забезпечення об'єднує компоненти системи до єдиної узгодженої системи із бажаними функціями. Етап інтеграції повинен забезпечити функціональну цілісність системи шляхом реалізації відповідної взаємодії та координації підсистем. Інтеграцію програм AI/ML можна розглядати як двоетапний процес: спочатку інтеграція компонентів підсистеми AI/ML, а потім інтеграція підсистем AI/ML з іншими підсистемами цільової системи. Тож визначений інтерфейс між компонентами AI/ML та іншими компонентами може впливати на процес і складність фази інтеграції. Нагадаємо, що моделі AI/ML постійно розвиватимуться. Тобто робочий процес AI/ML повинен сприяти безперервній інтеграції моделей AI/ML.

Тому на фазі інтеграції слід враховувати дві рекомендації:

– інтеграція об'єднує компоненти системи з метою забезпечення функціональної цілісності системи;

– процес інтеграції має відповідати конструктивним міркуванням, які зазначені в проєкті системи.

Фаза 5: тестування [6]. Оскільки результати моделей машинного навчання є стохастичними за своєю природою, виникають проблеми з детермінованими результатами для порівняння та перевірки програм машинного навчання.Отже, наявні фреймворки модульного тестування не можуть бути використані для тестування програм AI/ML. До того ж моделі AI/ML навчаються на вхідних даних адаптивним та ітеративним способом і залежать від багатьох параметрів, таких як вибрані функції, архітектура моделі та навіть дані навчання. Правила, створені системами AI/ML, можуть бути навіть невідомі розробникам. Усе це суттєво ускладнює виявлення помилкової поведінки системи та точне встановлення джерела помилок.

Слід також враховувати, що алгоритми AI/ML іноді можуть виявляти стійкість до деяких помилок і давати прийнятні результати шляхом компенсації зашумлених даних або помилок реалізації. Це також ускладнює виявлення та виправлення помилок в системі AI/ML. Тестування програмних систем AI/ML може потребувати великих навчальних даних, а ручне маркування таких даних є дорогим. Крім того, випадковий вибір підмножин даних, швидше за все, не зможе визначити багато граничних випадків. Усі ці проблеми роблять тестування та виправлення помилкової поведінки в системах AI/ML дуже складною задачею.

На фазі тестування слід брати до уваги такі рекомендації.

– Моделі AI/ML непрозорі, і їхню поведінку важко пояснити, тому вони потребують ретельного тестування в широкому діапазоні параметрів і для всіх можливих сценаріїв використання.

– Помилки в програмній системі AI/ML важко виявити через стохастичний характер моделей AI/ML. Тому всі підсистеми – як підсистеми з елементами AI/ML, так і решта підсистем –маютьі бути ретельно протестовані як на рівні модульного тестування, так і на рівні інтеграційного тестування.

Фаза 6: розгортання [7]. Фаза розгортання запускає програмну систему до експлуатації. Як правило, фаза розгортання оновлює або замінює наявну систему. Для програм машинного навчання протягом цієї фази виникає потреба в додаванні нових функціональних модулів до наявної системи. Однією з важливих проблем, які слід враховувати під час розгортання системи AI/ML, є те, що платформа та інфраструктура для експлуатації системи можуть суттєво відрізнятися від середовища, в якому була навчена та оцінена модель AI/ML. Ці відмінності можуть викликати проблеми сумісності, переносимості та масштабованості, а також можуть вплинути на продуктивність системи.

Рекомендації, які слід брати до уваги на фазі розгортання, очевидні:

– під час розгортання слід враховувати відмінності платформ в середовищах розробки та експлуатації;

– розгортання систем AI/ML в експлуатаційному середовищі має враховувати вимоги до переносимості та масштабованості;

– стратегія та реалізація розгортання мають бути достатньо обережними, щоб не вплинути на користувачів системи.

Проблеми та рекомендації машинного навчання

Відмінні характеристики та вимоги програмних систем AI/ML потребують чітко визначеного набору принципів і рекомендацій.

Формулювання задачі. Алгоритми машинного навчання пропонують рішення загального призначення. Щоби визначити адекватний алгоритм необхідно правильно сформулювати задачу, адже неправильне формулювання задачі може призвести до принципової неможливості програмної системи AI/ML працювати коректно. Правильна постановка задачі є передумовою успіху інших етапів розробки систем AI/ML.

Формулюючи задачі, слід брати до уваги наступні міркування:

– Правильне формулювання задачі системи як задачі машинного навчання є необхідною умовою успіху програмних систем AI/ML. Формулювання задачі машинного навчання має базуватися на чіткому розумінні цільових результатів та характеристик даних.

– Алгоритми машинного навчання пропонують рішення загального призначення. Отже, розробники повинні розуміти алгоритми, щоб вибрати відповідний алгоритм для конкретної задачі.

Збір даних. Збір і обробка великого обсягу даних є критичними накладними витратами для машинного навчання. Недостатня кількість даних також є проблемою для програмних систем AI/ML. Збір даних має бути зосереджений на повноті (представництві повного діапазону поведінки), точності (правильність даних), узгодженості (відсутність суперечливих даних) і своєчасності (відповідності поточному стану системи) даних для забезпечення якості даних. Проте підтримка якості даних вимагає ретельного збору, контролю та обслуговування даних. Це часто дуже дорого з огляду на час і пов'язані з цим обчислювальні роботи.

Отже, щодо збору даних слід враховувати наступне:

– процес збору даних має забезпечувати повноту, точність, послідовність і своєчасність набору даних;

– структура даних може змінюватися з часом, процес збору даних має бути гнучким, щоб легко адаптувати зміни в структурі та організації даних;

– вимоги до даних мають бути проаналізовані до отримання великого обсягу даних, що може заощадити значний час і ресурси в робочому процесі машинного навчання.

Попередня обробка. Необроблені дані можуть бути непридатними для використання в моделях машинного навчання, тому потребують попередньої обробки для очищення, організації та, за потреби, трансформації. «Брудні» дані вважаються головною проблемою для алгоритмів машинного навчання, оскільки можуть негативно впливати на навчання та отримані результати. Попередня обробка даних є важливою та складною фазою для машинного навчання і може потребувати значних витрат часу та зусиль.

Таким чином слід взяти до уваги дві рекомендації.

– Для систем, які керовані даними, правилом є «сміття на вході – сміття на виході» (garbage in – garbage out). Тому необроблені дані потребують попередньої обробки для видалення шуму, заповнення відсутніх значень та виконання інших потрібних перетворень.

– Необхідна попередня обробка може бути спільною для кількох елементів AI/ML. Тобто загальний робочий процес має максимізувати повторне використання попередньо оброблених даних.

Виділення ознак. Виділення ознак виконує оптимальне перетворення вхідних даних у вектори ознак для алгоритмів машинного навчання. Ця фаза виділяє набір ознак, які найкраще представляють приховані характеристики даних для машинного навчання і спрямоване на усунення потенційних шумів і надмірності з даних. Отримання низькорозмірного подання даних збільшує швидкість навчання та дозволяє візуалізувати результати роботи алгоритмів машинного навчання. Проте некоректно обрані ознаки можуть суттєво погіршити результати, що їх отримують за допомогою реалізованих моделей.

Отже, при виділенні ознак слід керуватися такими рекомендаціями.

– Якість набору ознак суттєво впливає на продуктивність моделей машинного навчання. Таким чином, розробники повинні знайти найкращий набір ознак.

– Слід створити автоматизований конвеєр для виділення ознак, оскільки цей процес вимагає часу та ресурсів.

– «Брудні» ознаки можуть негативно впливати на продуктивність моделі. Отже, ознаки слід очистити від забруднень.

– Кількість виділених ознак пов'язана зі складністю моделей. Тому слід намагатися виділяти найменший набір ознак, достатній для збереження прихованих в даних властивостей.

– Необхідно періодично переглядати набори ознак.

Побудова моделі. Моделі AI/ML створюються на основі конкретних алгоритмів машинного навчання залежно від задачі та характеристик даних. Можна використовувати наявні моделі з бібліотек або створювати власні моделі. Потім моделі ітераційно навчаються, доки не досягнуть бажаного рівня продуктивності. Однією з поширених проблем для моделей AI/ML є «перенавчання». У цьому випадку модель добре працює на навчальному наборі даних, але погано узагальнюється на інші дані. Причиною цього може бути занадто складна модель. Тому рішення полягає в знаходженні найпростішої моделі, яка забезпечує потрібну продуктивність. Проте занадто прості моделі машинного навчання, швидше за все, будуть недостатні і не зможуть зафіксувати приховані закономірності в даних. Також слід враховувати те, що розподіл даних має бути збалансованим, інакше висновок моделі може зміщуватися в бік класу, що домінує в навчальних даних.

Таким чином, основними рекомендаціями щодо побудови моделі є:

– алгоритми машинного навчання слід адаптувати відповідно до задачі та характеристик даних;

– розробники моделей машинного навчання мають починати з найпростішого рішення і поступово переходити до складніших рішень, враховуючи компроміс між ресурсами та продуктивністю;

– розробники мають забезпечити якість і збалансований розподіл даних;

– по змозі, розробникам слід розглянути можливість повторного використання наявних рішень (для підвищення продуктивності), перш ніж розробляти індивідуальні або більш інноваційні рішення.

Оцінка. Моделі машинного навчання оцінюються шляхом їх застосування до верифікаційного набору даних, який відокремлено від набору даних для навчання. Важливим є не лише оцінювання перед розгортанням, а й моніторинг продуктивності після розгортання, оскільки з часом продуктивність може знизитися через зміни в характеристиках вхідних даних. Тобто моделі машинного навчання можуть потребувати оновлення (наприклад, перенавчання), щоб адаптуватися до змін. Тому оцінка моделей машинного навчання може бути ітеративною протягом життєвого циклу. Крім того, в програмній системі AI/ML можуть застосовуватися різні взаємодіючі моделі. Отже, продуктивність окремих моделей може представляти лише частину наскрізних сценаріїв. Таким чином, важливо мати оцінки як на рівні окремих моделей, так і на рівні всієї системи.

Рекомендації, яких слід дотримуватися щодо оцінювання, наступні.

– Набір верифікаційних даних має бути достатньо повним, щоб представляти всі можливі сценарії використання.

– Оцінка моделі повинна зосереджуватися не лише на точності та інших характеристиках моделі, а і брати до уваги такі характеристики, як пропускна здатність, використання ресурсів і масштабованість.

– Коли різні моделі взаємодіють, слід оцінювати продуктивність як на рівні моделі, так і на рівні системи.

Інтеграція та розгортання моделі. Наступним кроком в життєвому циклі для навчених моделей є їх інтеграція до цільової системи, яка передбачає об'єднання всіх необхідних компонентів (наприклад, моделей, конвеєрів вводу-виводу). Для кількох моделей може знадобитися визначення та реалізація інтерфейсів для кожної моделі, щоб забезпечити їхню взаємодію з іншими моделями та компонентами. Одним із поширених підходів до розгортання моделей машинного навчання є розгортання їх як сервісів (service) і доступ до сервісів через API. Розгортання моделей ML має враховувати портативність і сумісність моделей щодо цільової платформи.

Під час інтеграції та розгортання моделі слід керуватися такими рекомендаціями:

– інтеграція компонентів машинного навчання має забезпечувати функціональну цілісність програмних систем AI/ML;

– під час розгортання слід враховувати сумісність і відмінності в платформах розробки та експлуатації;

– розгортання має забезпечувати плавні зміни наявної системи без впливу на користувача чи бізнес-процеси.

Управління моделлю. Управління моделлю машинного навчання, яке включає її навчання, підтримку, розгортання, моніторинг і документування, є складним завданням у робочому процесі. Моделі машинного навчання керуються даними та базуються на різних припущеннях щодо розподілу та шаблонів даних. Однак початкові характеристики даних можуть не зберігатися через зміни в даних, що з великою ймовірністю вплине на поведінку моделі. Таким чином, важливо контролювати продуктивність розгорнутих моделей, відстежувати зміни в характеристиках даних, а також перенавчати та повторно перевіряти моделі. Для цього потрібні ітерації життєвого циклу моделі машинного навчання, що зазвичай вимагає великих витрат часу та ресурсів.

Також важливо, хоч і складно забезпечити відстеження версій моделі, наборів даних і конфігурації, щоб забезпечити відтворюваність моделей машинного навчання і полегшити керування робочим процесом. Відтворюваність моделі допомагає аналізувати та порівнювати поведінку та продуктивність моделі, а також підтримує рішення щодо розгортання.

Відповідно щодо управління моделлю слід дотримуватися таких рекомендацій.

– Моніторинг моделей після розгортання є важливим. Для забезпечення потрібного рівня продуктивності моделі можуть потребувати перенавчання, що може ініціювати цикл технічного обслуговування.

– Дані необхідно відстежувати, оскільки характеристики та розподіл даних із часом можуть змінюватися.

– Побудова моделі з бажаною продуктивністю вимагає дослідження та експериментування. Версії моделей, даних і конфігурацій важливі для полегшення відтворюваності моделей машинного навчання.

– Кожна фаза життєвого циклу підсистеми AI/ML повинна бути добре задокументована для забезпечення підтримки моделі.

Етика в розробці AI/ML. Для уникнення негативних наслідків впровадження елементів AI/ML в різні сфери застосування програмного забезпечення, дуже важливо переконатися, що використання AI або ML відповідає етичним нормам. Дослідники та практики повинні «відповідально» використовувати AI. Командам, які беруть участь у дослідженні та розробці програмної системи AI/ML, сліді дотримуватися стандартного кодексу етики програмної інженерії [8].

Загалом при розробці необхідно притримуватися таких рекомендацій щодо етики в розробці AI:

– розробка програмних систем AI/ML має відповідати принципам етики, щоб забезпечити відповідальне використання AI;

– необхідно зберегти конфіденційність і безпеку особистих і бізнесових даних;

– при використанні AI/ML колективний добробут має бути пріоритетнішим за бізнес-вигоди.

Висновки

Програмне забезпечення вже давно є невід'ємною складовою нашого життя. Останнім часом спостерігається бурхливе зростання інтересу та застосування в різних бізнесових сферах програмних систем з елементами штучного інтелекту. Життєвий цикл, моделі та технології реалізації елементів AI/ML є досить специфічними, і це має бути враховано в процесі інженерії програмних систем для забезпечення їхньої продуктивності та відповідності бізнес-потребам.

Життєвий цикл розробки програмного забезпечення наразі є добре формалізованим та супроводжується визнаними добірками рекомендацій та найкращих практик. Проте у випадку розробки програмних систем з елементами AI/ML ці рекомендації потребують відповідних модифікацій.

В цій роботі проаналізовано особливості розробки систем з елементами AI/ML з двох точок зору:

– з точки зору інженерії програмного забезпечення проаналізовані зміни, які слід брати до уваги у розробці програмних систем з елементами AI/ML;

– з точки зору розробки моделей машинного навчання проаналізовано фактори, які слід брати до уваги при програмній реалізації моделей, щоб зробити процес програмної реалізації керованим та передбачуваним.

Практичну корисність роботи обумовлено добірками рекомендацій, які сформульовано за кожною проаналізованою проблемою. Наступним етапом цього дослідження є складання докладних системних рекомендацій до процесу розробки програмного забезпечення з елементами AI/ML.

Роботу виконано за фінансової підтримки програми «Hamburg Program for Scholars at Risk – Science Bridge for Ukraine», що фінансується Міністерством науки, досліджень, рівності та округів (Ministry of Science, Research, Equalities and Districts) м. Гамбург (Німеччина).

Література

1. 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.
2. Alvarez-Rodríguez J. M., Zuñiga 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.
3. 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.
4. Washizaki H., Khomh F., Guéhéneuc 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.
5. Martínez-Fernández 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.
6. 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.
7. 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: 14.08.2022].
8. 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: 14.08.2022].

References

1. CHALLA H., NIU N. & JOHNSON R. (2020) Faulty Requirements Made Valuable: On the Role of Data Quality in Deep Learning. *IEEE 7th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pp. 61–69. Available from: https://doi.org/10.1109/AIRE51212.2020.00016 [Accessed 08/13/2022].
2. ALVAREZ-RODRÍGUEZ J. M. et al. (2019) Challenges and opportunities in the integration of the Systems Engineering process and the AI/ML model lifecycle. *INCOSE International Symposium*. Vol. 29, No. 1, pp. 560–575. Available from: https://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2019.00621.x [Accessed 08/13/2022].

3. VOGELSANG A. & BORG M. (2019) Requirements Engineering for Machine Learning: Perspectives from Data Scientists. *IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pp. 245-251. Available from: https://doi.org/10.1109/REW.2019.00050 [Accessed 08/13/2022].
4. WASHIZAKI H. et al. (2022) Software-Engineering Design Patterns for Machine Learning Applications. *Computer*. 55(3), pp. 30-39. Available from: https://doi.org/10.1109/MC.2021.3137227 [Accessed 08/13/2022].
5. MARTÍNEZ-FERNÁNDEZ S. et al. (2022) Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology*. 31(2), Article 37e. Available from: https://doi.org/10.1145/3487043 [Accessed 08/13/2022].
6. NISHI Y., MASUDA S., OGAWA H & UETSUKI K. (2018) A Test Architecture for Machine Learning Product. *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 2018. P. 273-278, Available from: https://doi.org/10.1109/ICSTW.2018.00060 [Accessed 08/13/2022].
7. PALEYES A., URMA R.-G. & LAWRENCE N. D. (2022) Challenges in Deploying Machine Learning: a Survey of Case Studies. *ACM Computing Surveys*. Available from: https://dl.acm.org/doi/10.1145/3533378 [Accessed 08/13/2022].
8. Code of Ethics. IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices. Available from: https://www.computer.org/education/code-of-ethics [Accessed 08/14/2022].

Одержано 17.08.2022

Про автора:

Любченко Віра Вікторівна,
доктор технічних наук,
професор кафедри інженерії програмного забезпечення
Кількість публікацій у вітчизняних виданнях: 52.
Кількість публікацій у закордонних виданнях: 5.
http://orcid.org/0000-0002-4611-7832.

Місце роботи автора:

Національний університет «Одеська політехніка», 65044, Одеса,
проспект Шевченка, 1.
Тел.: +38 048 705 8566
E-mail: lvv@op.edu.ua

Прізвища та ім’я авторів і назва доповіді англійською мовою:
Liubchenko V. V.
Some aspects of software engineering for AI-based systems

Прізвища та ім’я авторів і назва доповіді українською мовою:
Любченко В. В.
Про деякі аспекти інженерії програмних систем
з елементами штучного інтелекту

Контакти для редактора: Любченко Віра Вікторівна,
професор кафедри інженерії
програмного забезпечення Національного університету
«Одеська політехніка»,
e-mail: lvv@op.edu.ua, тел.: +38 050 392 2150