УДК 94:061.213(477)»19/20» DOI https://doi.org/10.35546/kntu2078-4481.2025.3.2.10

В. Г. ВАСЕНКО
студент кафедри комп'ютерні науки
Херсонський національний технічний університет
ORCID: 0009-0003-2558-2588

Н. В. КОРНІЛОВСЬКА
кандидат технічних наук,
доцент кафедри інформатики і комп'ютерних наук
Херсонський національний технічний університет
ORCID: 0000-0002-8331-8027

С. В. ВИШЕМИРСЬКА
кандидат технічних наук,
доцент кафедри інформатики і комп'ютерних наук
Херсонський національний технічний університет
ORCID: 0000-0002-6343-7512

М. В. КАРАМУШКА
кандидат технічних наук,
доцент кафедри інформатики і комп'ютерних наук
Херсонський національний технічний університет
ORCID: 0000-0001-5982-4598

ЗАСТОСУВАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ СТВОРЕННЯ УТИЛІТ ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS

У сучасному світі, Windows продовжує залишатися однією з найпопулярніших операційних систем для користувачів. Проте, при тривалому користуванні цією системою, люди часто зіткнюються з проблемами щодо швидкодії та накопиченням непотрібних файлів, що може вплинути на продуктивність комп'ютера. Засоби для очищення системи, які використовуються традиційно, не завжди враховують індивідуальні потреби користувачів, що може призвести до втрати важливих даних або недостатньо ефективного очищення. Зростання вимог до точності і безпеки процесу очищення підкреслює значення використання методів машинного навчання для автоматизації цього процесу.

У цій роботі досліджується розробка та впровадження інтелектуальної утиліти для очищення та оптимізації операційної системи Windows з використанням методів машинного навчання на основі бібліотеки ML.NET. Програма створена на мові C# із застосуванням WinUI 3 для побудови сучасного користувацького інтерфейсу та PowerShell для виконання системної оптимізації. Технологія машинного навчання дозволяє ефективно класифікувати файли за ознаками (тип, розмір, дата створення, дата змінення тощо) та визначати, чи є вони потенційно непотрібними. Розроблена програма пропонує як стандартний функціонал очищення за категоріями, так і функцію інтелектуального очищення, яка мінімізує ризик видалення важливих даних.

Для оцінки ефективності запропонованого рішення проведено серію експериментів з класифікації файлів різних категорій та порівняльний аналіз з вбудованими у Windows утилітами. Експерименти включали тестування точності моделі машинного навчання на наборі даних, що містить файли різних типів та характеристик. Дослідження демонструє значні переваги інтелектуального підходу, що дозволяє автоматизувати процес очищення системи та зменшити кількість помилок.

Дане дослідження має значну практичну цінність для розробників програмного забезпечення, які працюють над інструментами для очищення та оптимізації операційних систем. Воно демонструє, як сучасні методи машинного навчання можуть підвищити точність та безпеку очищення, відкриваючи шлях до більш інтелектуальних та зручних утиліт для підтримки продуктивності комп'ютерних систем.

Ключові слова: *машинне навчання, очищення системи, ML.NET, WinUI 3, C#, класифікація файлів, оптимізація Windows, PowerShell, інтелектуальна утиліта.*

V. G. VASENKO
Student at the Department of Computer Science
Kherson National Technical University
ORCID: 0009-0003-2558-2588

N. V. KORNILOVSKA
Candidate of Technical Sciences,
Associate Professor at the Department of Computer Science
Kherson National Technical University
ORCID: 0000-0002-8331-8027

S. V. VYSHEMYRSKA
Candidate of Technical Sciences,
Associate Professor at the Department of Computer Science
Kherson National Technical University
ORCID: 0000-0002-6343-7512

M. V. KARAMUSHKA
Candidate of Technical Sciences,
Associate Professor at the Department of Computer Science
Kherson National Technical University
ORCID: 0000-0001-5982-4598

APPLICATION OF MACHINE LEARNING METHODS FOR CREATING WINDOWS OPERATING SYSTEM UTILITIES

In the modern world, Windows continues to remain one of the most popular operating systems for users. However, with prolonged use of this system, people often encounter problems regarding performance and accumulation of unnecessary files, which can affect computer productivity. Traditional system cleaning tools do not always consider individual user needs, which can lead to loss of important data or insufficiently effective cleaning. The growing requirements for accuracy and safety of the cleaning process emphasize the importance of using machine learning methods to automate this process.

This work investigates the development and implementation of an intelligent utility for cleaning and optimization of the Windows operating system using machine learning methods based on the ML.NET library. The program is created in C# language with the application of WinUI 3 for building a modern user interface and PowerShell for performing system optimization. Machine learning technology allows effective classification of files by features (type, size, creation date, modification date, etc.) and determines whether they are potentially unnecessary. The developed program offers both standard cleaning functionality by categories and an intelligent cleaning function that minimizes the risk of deleting important data.

To evaluate the effectiveness of the proposed solution, a series of experiments on file classification of various categories and comparative analysis with built-in Windows utilities were conducted. The experiments included testing the accuracy of the machine learning model on a dataset containing files of different types and characteristics. The research demonstrates significant advantages of the intelligent approach, which allows automating the system cleaning process and reducing the number of errors.

This research has significant practical value for software developers working on tools for cleaning and optimization of operating systems. It demonstrates how modern machine learning methods can improve the accuracy and safety of cleaning, opening the path to more intelligent and convenient utilities for maintaining computer system productivity.

Key words: *machine learning, system cleaning, ML.NET, WinUI 3, C#, file classification, Windows optimization, PowerShell, intelligent utility.*

Постановка проблеми

Сучасні операційні системи Windows накопичують надлишкові файли, що знижує продуктивність. Традиційні утиліти очищення не використовують машинне навчання для безпечної класифікації файлів.

Формулювання мети дослідження

Метою дослідження є розробка інтелектуальної утиліти для очищення та оптимізації операційної системи Windows з використанням методів машинного навчання на основі бібліотеки машинного навчання ML.NET та порівняння її ефективності з традиційними підходами до очищення системи.

Аналіз останніх досліджень і публікацій

Аналіз існуючих рішень показує, що сучасні утиліти очищення Windows, такі як BleachBit, базуються на статичних правилах категоризації файлів без можливості вибіркового видалення. Системні оптимізатори на зразок Chris Titus Windows Utility забезпечують широкий функціонал PowerShell-команд, однак мають обмежену прозорість процесів виконання.

Дослідження ефективності ML.NET для класифікації файлів демонструють можливість досягнення точності понад 80 % при використанні логістичної регресії та наївного баєсівського класифікатора. Однак відсутні комплексні рішення, що поєднують інтелектуальне очищення з машинним навчанням та системну оптимізацію в єдиному інтерфейсі.

Для досягнення поставленої мети виконано такі завдання: розроблено програмне забезпечення з графічним інтерфейсом, яке підтримує стандартний функціонал очищення системи та інтелектуальний режим на основі ML.NET; реалізовано класифікацію файлів за ознаками; проведено тестування точності моделі машинного навчання.

Викладення основного матеріалу дослідження

У технологічному стеку утиліти є такі компоненти:

- .NET – основа платформи розробки;
- C# – мова програмування для логіки додатку;
- WinUI 3 – платформа для графічного інтерфейсу Windows;
- Python – для створення моделей класифікації [1];
- ML.NET – фреймворк машинного навчання для інтелектуального очищення.

Рис. 1. Схема використаного технологічного стеку

Розроблена програмна система складається з чотирьох підпроєктів:

- OptimizeUtility – головний застосунок з графічним інтерфейсом на базі WinUI 3 та C#;
- OptimizeUtility.Core – бібліотека класів для операцій з файлами;
- OptimizeUtility.ModelCreator – Python-застосунок для створення моделей машинного навчання у форматі ONNX [1];
- OptimizeUtility.ScriptRunner – компонент для виконання PowerShell-скриптів.

Основою інтерфейсу є ShellPage.xaml – базова структурна «обгортка» застосунку, що реалізує навігаційний механізм. Структура включає:

1. AppTitleBar – кастомна верхня панель з логотипом та стандартними кнопками керування вікном.
2. NavigationView – вертикальна навігаційна панель з основними сторінками.

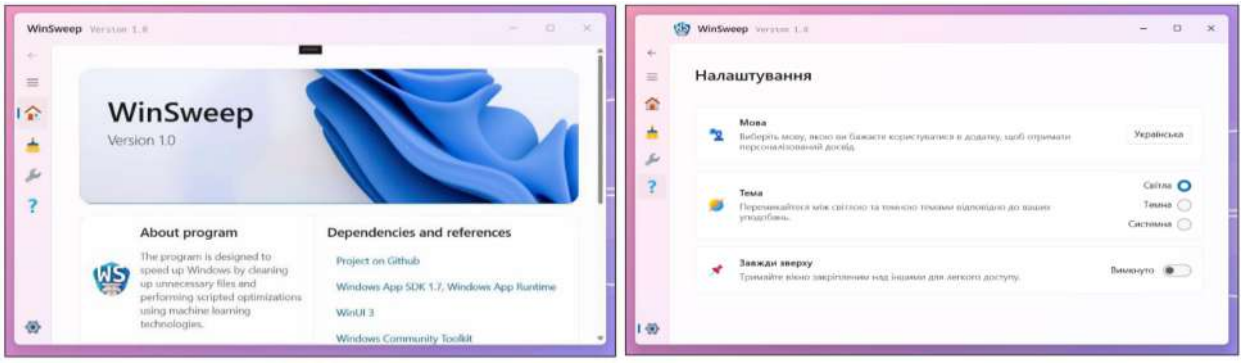

Рис. 2. Демонстрація головної сторінки програми та сторінки налаштувань

Сторінка очищення відповідає за стандартне та інтелектуальне очищення системи.

Стандартний режим очищення містить двоколонкову сітку: ліва частина – чекбокси для вибору категорій файлів, права частина – таблиця DataGrid з деталями файлів (чекбокси вибору, шлях, розмір, категорія). Під таблицею розташована панель керування з кнопками для запуску сканування по категоріях, видалення файлів, скасування сканування, активації/деактивації всіх чекбоксів.

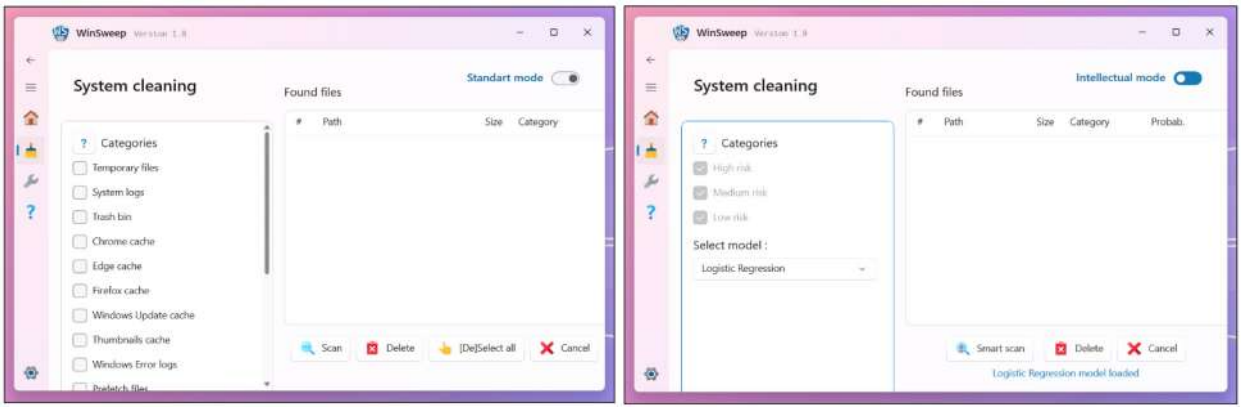

Рис. 3. Демонстрація сторінок очищення системи: ліворуч – стандартне; праворуч – інтелектуальне

Інтелектуальний режим активується через перемикач ToggleSwitch і має розширений функціонал порівняно зі стандартним режимом. Структура інтерфейсу подібна до StandardMode, але з додатковими можливостями машинного навчання.

Заголовки колонок містять додатковий стовпець «Вірогідність», який показує ймовірність того, що файл є надлишковим. Зліва додані чекбокси для фільтрації за вірогідністю та випадаючий список вибору моделі класифікації.

Ключовою особливістю SmartMode є колірне маркування рядків у таблиці результатів:

- Зелений – висока вірогідність, що файл непотрібний.
- Жовтий – середня вірогідність.
- Червоний – низька вірогідність.

Така візуалізація допомагає користувачеві швидко оцінити, які файли варто видалити, а які залишити, базуючись на результатах аналізу машинного навчання.

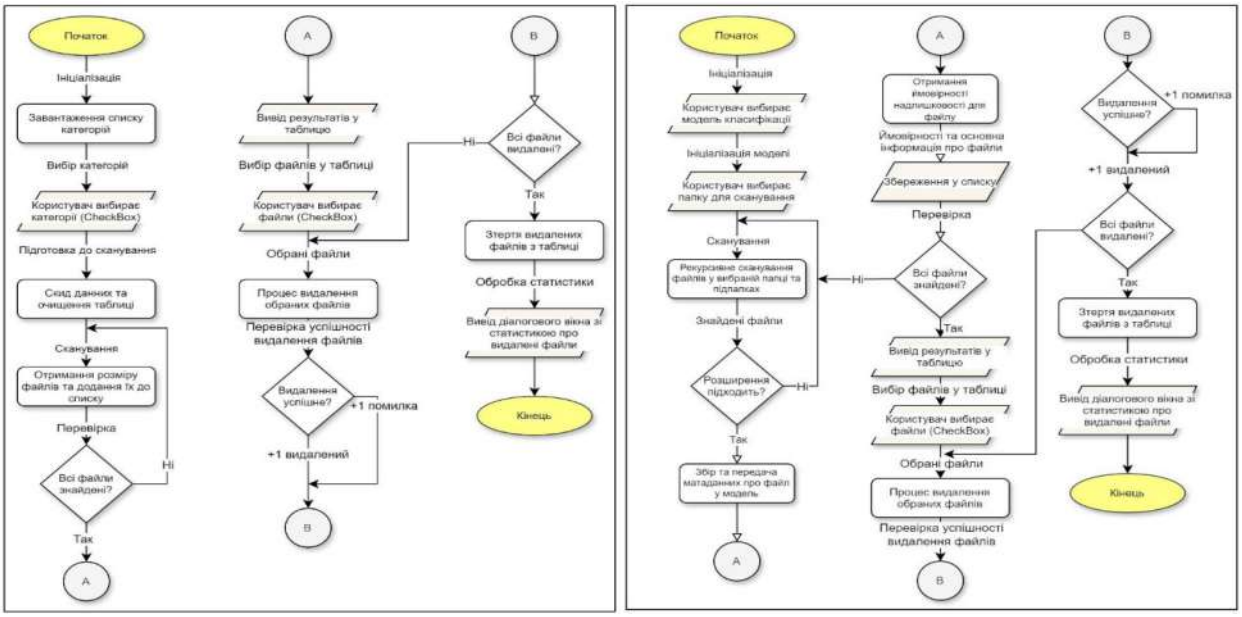

Рис. 4. Схеми очищення системи: ліворуч – стандартне; праворуч – інтелектуальне

Далі в нас йде наступний екран – це екран оптимізації системи або OptimizationPage, об'єктну схему якого можна побачити на рис. 4. Загалом принцип тут дещо схожий на попередній інтерфейс. Ми маємо ScrollViewer, в якому знаходиться певна кількість кнопок, які відповідають за додавання до текстового поля TextBox певних команд.

Після цього користувачу доступні три кнопки для взаємодії:

- Кнопка виконання записаних у текстове поле скриптів/команд.
- Кнопка очищення текстового поля від будь-яких записаних команд.
- Кнопка збереження вмісту текстового поля як файлу виконання PowerShell скриптів (тип файлу.ps1).

Тестування інтелектуального очищення

Для експериментів сформовано набір даних з 7 ознак (Extension, SizeMB, AgeInDays, LastModifiedDays, IsTemporary, AccessFrequency) та цільовою змінною IsJunk. Числові ознаки стандартизовано, категоріальні перетворено через OneHotEncoder [2].

Розподіл цільової змінної збалансований: 60 % не сміттєвих файлів, 40 % сміттєвих. Для логістичної регресії використано GridSearchCV з крос-валідацією, наївний баєсівський класифікатор – з параметрами за замовчуванням [3].

Середні значення F1-міри за крос-валідацією:

- Логістична регресія: 0.7908 (±0.0480);
- Наївний баєсівський класифікатор: 0.7805 (±0.0288).

Точність на тестовій вибірці (1000 записів):

- Логістична регресія: 0.8320;
- Наївний баєсівський класифікатор: 0.8190.

Обидві моделі продемонстрували високу точність та стабільність метрик. Незначні розбіжності між результатами валідаційної та тестової вибірок можуть свідчити про початкові ознаки перенавчання, що потребує подальшого дослідження впливу мультиколінеарності ознак та розширення навчального датасету.

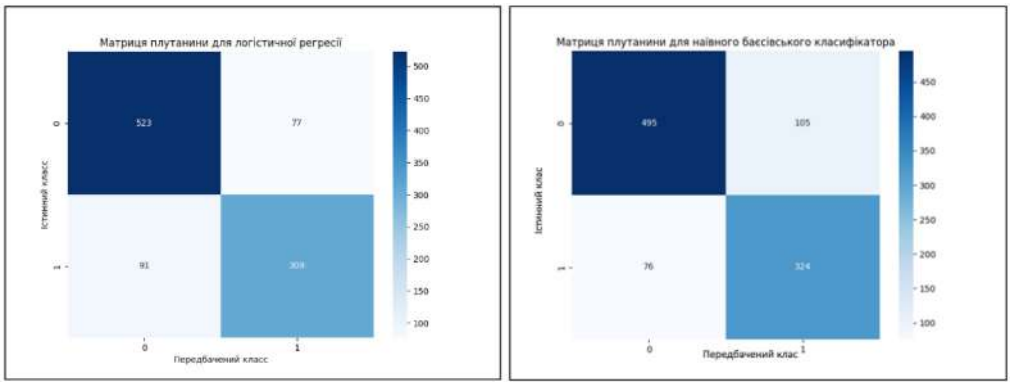

Рис. 5. Матриці плутанини: ліворуч – для логістичної регресії; праворуч – для наївного баєсівського класифікатора

Тестування стандартного очищення

Початкове тестування розпочалося з перевірки функції стандартного очищення. Під час розробки програми накопичилась значна кількість зайвих файлів у системі. Зазвичай після ручного очищення обсяг вільного місця на системному диску становить приблизно 42–46 ГБ.

Було проведено сканування з активацією всіх категорій пошуку. При стандартному запуску система ідентифікувала 4,6 ГБ надлишкових файлів, однак аналіз виявив обмеження через відсутність адміністративних прав. Після запуску з підвищеними правами повторне сканування виявило 23,4 ГБ надлишкових файлів. Реалізована функція масового перемикання чекбоксів дозволяє зручно обирати необхідні об’єкти для видалення.

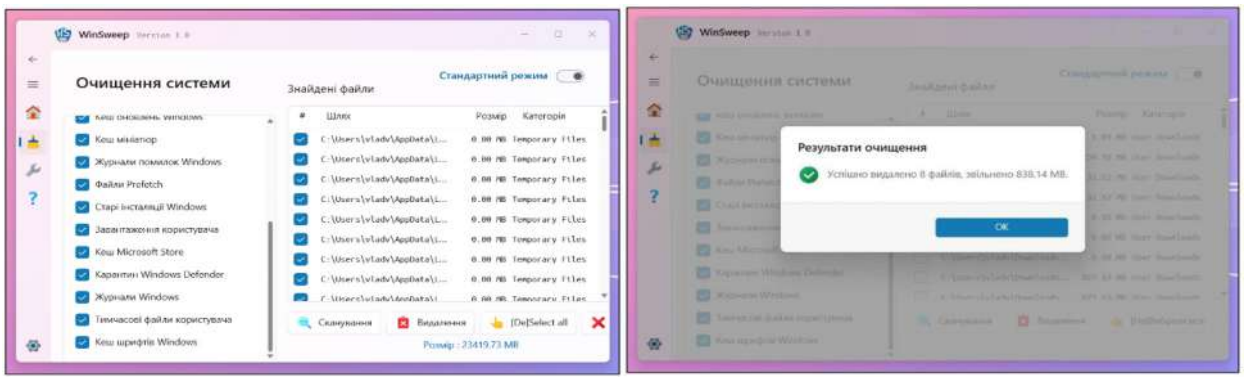

Рис. 6. Тестування стандартного очищення : ліворуч – знайденні сміттєві файли 23Гб; праворуч – результат часткового очищення

Тестування оптимізації системи

Для оцінювання ефективності оптимізації системи за допомогою PowerShell-команд проведено експеримент у віртуальній машині з «чистою» Windows 11.

Алгоритм тестування: спочатку фіксувалися базові показники системи через «Диспетчер задач» протягом 10 секунд, потім застосовувалися оптимізаційні скрипти з подальшим перезавантаженням та повторним заміром. Програмний модуль реалізує 15 команд оптимізації: перевірка цілісності системних файлів, очищення DNS-кешу, вимкнення телеметрії, переведення сервісів у ручний режим запуску, очищення браузера Edge та вимкнення фонових процесів.

- До оптимізації: CPU – 7 %, пам'ять – 2,7/5,7 GB (47 %), процесів – 154, потоків – 2117, хендлів – 6016.
- Після оптимізації: CPU – 4 %, пам'ять – 2,0/5,7 GB (35 %), процесів – 99, потоків – 1869, хендлів – 4877.

Графік процесора став стабільнішим без різких стрибків.

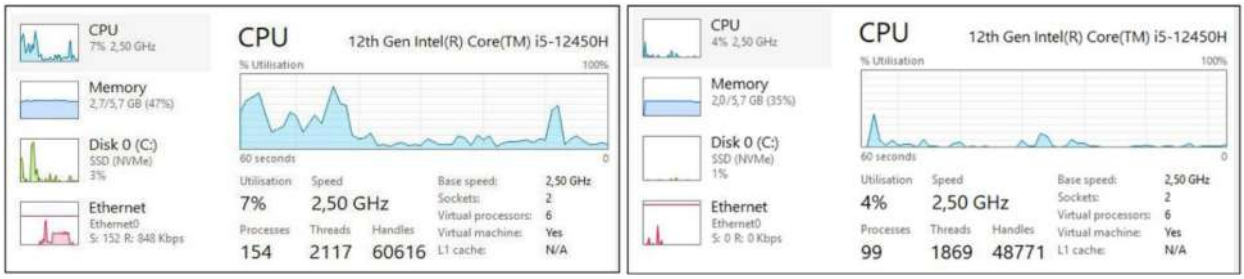

Рис. 7. Заміри системи: ліворуч – до оптимізації; праворуч – після оптимізації

Порівняння показників підтверджує ефективність оптимізаційних скриптів, які значно знизили фонове навантаження на систему та забезпечили стабільне покращення роботи ОС.

Висновки

Розроблена інтелектуальна утиліта для очищення та оптимізації операційної системи Windows з використанням методів машинного навчання демонструє суттєві переваги порівняно з традиційними підходами. Експериментальні дослідження підтвердили ефективність застосування ML.NET для класифікації файлів, при цьому логістична регресія показала точність 83,20 %, а наївний баєсівський класифікатор – 81,90 %.

Практичне тестування системи виявило її високу ефективність: стандартний режим очищення дозволив ідентифікувати 23,4 ГБ надлишкових файлів, а модуль оптимізації забезпечив зниження навантаження на процесор з 7 % до 4 %, зменшення використання оперативної пам'яті з 47 % до 35 % та скорочення кількості фонових процесів з 154 до 99. Інтелектуальний режим очищення з візуальним маркуванням файлів за рівнем вірогідності значно підвищує безпеку процесу та мінімізує ризик видалення важливих даних.

Отримані результати мають значну практичну цінність для розробників системного програмного забезпечення та відкривають нові можливості для створення інтелектуальних утиліт оптимізації операційних систем. Розроблений підхід може служити основою для подальших досліджень у напрямку вдосконалення алгоритмів машинного навчання для системної аналітики, розширення функціоналу автоматизованого очищення та адаптації методики до інших операційних систем.

Список використаної літератури

1. Mueller, A. C., & Guido, S. (2021). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.
2. Bishop, C. M. (2023). Pattern Recognition and Machine Learning: Advances in Neural Networks. 2nd ed. Springer.
3. Звенигородський, О. С., Зінченко, О. В., Чичкарьов, Є. А., & Кисіль, Т. М. (2022). Штучний інтелект. Вступний курс: Навчальний посібник. Київ : ДУТ.

References

1. Mueller, A. C., & Guido, S. (2021). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.
2. Bishop, C. M. (2023). Pattern Recognition and Machine Learning: Advances in Neural Networks (2nd ed.). Springer.
3. Zvenigorodskyi, O. S., Zinchenko, O. V., Chychkaryov, Ye. A., & Kysil, T. M. (2022). Shtuchnyi intelekt. Vstupnyi kurs: Navchalnyi posibnyk [Artificial intelligence. Introductory course: Tutorial]. Kyiv : DUT. [in Ukrainian].

Дата першого надходження рукопису до видання: 20.09.2025
Дата прийнятого до друку рукопису після рецензування: 13.10.2025
Дата публікації: 28.11.2025