Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтелектуальна система прогнозування конфліктів програмного забезпечення у Windows
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-2
Дмитро Годін
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Євген Павленко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Годіну Дмитру Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система прогнозування конфліктів програмного забезпечення у Windows

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 червня 20 25 р.

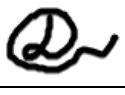
3. Вихідні дані до роботи наукові статті та публікації про методи прогнозування часових рядів; матеріали з інтернету, присвячені машинному навчанню та аналізу даних; документація та книги з Python, бібліотек pandas, NumPy, scikit-learn, TensorFlow

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Проектування системи
3) Програмна реалізіція

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05.2025	виконано
3	Дослідження існуючих аналогів	22.05.2025	виконано
4	Аналіз методів вирішення задачі	24.05.2025	виконано
5	Розробка системи	27.05.2025	виконано
6	Написання пояснювальної записки	28.05.2025	виконано
7	Перевірка на академічний плагіат	30.05.2025	виконано
8	Нормоконтроль	02.06.2025	виконано
9	Підготовка презентації та доповіді	03.06.2025	виконано
10	Попередній захист	05.06.2025	виконано
11	Рецензування	07.06.2025	виконано
12	Захист	19.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач  _____
(підпис)

Керівник роботи _____ доц. Євген Павленко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 64 с., 3 рис., 1 дод., 20 джерел.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ПРОГНОЗУВАННЯ КОНФЛІКТІВ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, PYTHON, PSUTIL, SCIKIT-LEARN, RANDOM FOREST.

Об’єктом дослідження є процес пониження продуктивності комп’ютера та виникнення конфліктів програмного забезпечення в операційній системі Windows.

Предметом дослідження є автоматизована інтелектуальна система прогнозування конфліктів програм, яка за допомогою штучного інтелекту завчасно повідомляє користувачів.

Метою роботи є створення зручного застосунку з використанням штучного інтелекту, який допоможе користувачам уникати повільної роботи комп’ютера.

Методами дослідження є аналіз взаємодії різних застосунків у оперативній системі Windows, аналіз використання ресурсів застосунками, методи машинного навчання для побудови прогностичних моделей, вивчення загальних потреб користувачів, методи проектування адаптивних користувацьких інтерфейсів.

Створений застосунок дозволяє значно підвищити стабільність роботи комп’ютерів на оперативній системі Windows, зменшити кількість випадків зависання та неочікуваного завершення програм, оптимізувати використання системних ресурсів та підвищити загальну продуктивність користувачів.

ABSTRACT

Bachelor’s thesis contains: 64 pp., 3 fig., 1 ann., 20 references.

INTELLIGENT SYSTEM, CONFLICT PREDICTION, SOFTWARE, PYTHON, PSUTIL, SCIKIT-LEARN, RANDOM FOREST.

The research object is the process of computer performance degradation and software conflicts occurrence in the Windows operating system.

The research subject is an automated intelligent system for predicting software conflicts that uses artificial intelligence to proactively notify users about potential issues.

The purpose of the study is to create a user-friendly application utilizing artificial intelligence to help users avoid slow computer performance.

The research methods include analysis of various applications’ interactions within the Windows operating system, analysis of resource utilization by applications, machine learning methods for building predictive models, studying general user needs, and methods for designing adaptive user interfaces.

The developed application enables significant improvement in Windows computer stability, reduces the number of system freezes and unexpected program terminations, optimizes system resource utilization, and enhances overall user productivity.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ARIMA – AutoRegressive Integrated Moving Average – статистична модель для аналізу та прогнозування часових рядів, яка поєднує авторегресію, інтегрування та ковзне середнє;

CPU – Central Processing Unit – центральний процесор;

CSV – Comma-Separated Values – текстовий формат даних, у якому поля розділені комами;

JSON – JavaScript Object Notation – легковаговий текстовий формат обміну даними у вигляді об’єктів JavaScript;

LSTM – Long Short-Term Memory – тип рекурентної нейронної мережі, здатний запам’ятовувати довгострокові залежності в послідовних даних;

MAE – Mean Absolute Error – середня абсолютна похибка;

PID – Process Identifier – унікальний ідентифікатор процесу в операційній системі;

RAM – Random Access Memory – оперативна пам’ять;

SARIMA – Seasonal Autoregressive Integrated Moving Average – розширення моделі ARIMA, що враховує сезонні компоненти в часових рядах;

SVM – Support Vector Machines – алгоритм машинного навчання для класифікації та регресії, який знаходить оптимальну гіперплощину, що розділяє дані різних класів з максимальним зазором.

ВСТУП

Сьогодні проблема конфліктів програмного забезпечення стає актуальнішою. Зі збільшенням кількості застосунків, які користувачі встановлюють на свої комп’ютери, зростає ймовірність виникнення конфліктів за системні ресурси. Це призводить до погіршення продуктивності системи, нестабільної роботи програм та, у результаті, негативного впливу на досвід користувачів. Особливо це стосується комп’ютерів із обмеженими технічними характеристиками, де ефективне керування ресурсами є критичним для забезпечення продуктивності роботи системи.

Статистичні дані показують, що у багатьох людей 1–20 % робочого часу витрачається на вирішення проблем із комп’ютером або через його повільну роботу [1]. При цьому більшість користувачів не мають достатніх технічних знань для виявлення причин цих проблем та їх вирішення. Згідно з дослідженням, середній користувач зустрічає одну проблему кожну годину роботи з комп’ютером [1].

Сучасні рішення для моніторингу системних ресурсів та оптимізації роботи комп’ютера базуються на реактивному підході, який передбачає реагування на проблеми, що вже виникли. Ця робота пропонує підхід, який відрізняється традиційних методів, намагаючись замість цього прогнозувати конфлікти. Також важливо, щоб застосунок був зрозумілим для всіх користувачів, тому потрібно приділити увагу розробці зрозумілому та зручному інтерфейсу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

У сучасній комп’ютерній екосистемі, де користувачі одночасно запускають багато різних програм, проблема конфліктів програмного забезпечення дедалі чіткіше виявляє себе на практиці. Збільшення кількості одночасно активних застосунків, яке спостерігається в умовах багатозадачності, спричиняє зростаюче навантаження на апаратні ресурси. Навіть при наявності потужних процесорів і великого обсягу оперативної пам’яті, багато користувачів не можуть дозволити собі оновлення комп’ютера до найсучасніших конфігурацій через фінансові обмеження. У результаті на звичайних домашніх та офісних комп’ютерах накопичуються десятки програм, встановлених для найрізноманітніших завдань: від офісного пакету та браузера з безліччю вкладок до фонових антивірусних сканувань і мультимедійних редакторів.

Сутність конфлікту програмного забезпечення полягає у суперечці процесів за одні й ті самі обмежені системні ресурси – центральний процесор, оперативну пам’ять, відеокарту, дисковий простір або мережеву пропускну здатність. Коли відбувається одночасне звернення різних програм до одного ресурсу, виникає ситуація, за якої система не може задовольнити всі запити в повному обсязі і вчасно. Найчастіше користувачі помічають це у вигляді загального «гальмування» інтерфейсу, затримок під час завантаження вікон та довших часів відкриття файлів, але при цьому не завжди правильно ідентифікують справжню причину сповільнення. Нерідко проблема списується на застарілий, хоча насправді проблема криється в недостатньо продуманому розподілі ресурсів і одночасному виконанні надмірно ресурсоємних завдань.

Протягом типового робочого дня користувач може зіткнутися з різними видами конфліктів програмного забезпечення.

Конфлікти за процесорний час – вони виникають, коли декілька інтенсивно обчислювальних застосунків – наприклад, відеоредактор, чи віртуальна машина – стартують одночасно або запускають фонові обчислення. Якщо одночасно з цим запускається антивірусне сканування або різні автоматичні оновлення, сумарне навантаження на ядра ЦП може досягнути 100 %, що призведе до значних затримок не лише у важких програмах, але й у простих завданнях, таких як введення тексту чи перемикання між вікнами.

Конфлікти за оперативну пам’ять – пам’ять є одним із найдорожчих ресурсів у системі: нестача вільної RAM призводить до активного звернення до файлу підкачки на жорсткому диску або SSD, що на багато разів повільніше. Також, у разі відкриття великої кількості вкладок у браузері, одночасного запуску декількох віртуальних машин чи ємкісних наукових обчислень система починає переміщати дані між пам’яттю та накопичувачем, створюючи помітні затримки у всіх програмах. Прогнозування конфліктів оперативної пам’яті є найголовнішим, серед інших ресурсів.

Конфлікти за графічні ресурси – графічний процесор стає активно використовуватись, коли користувачі одночасно працюють із вимогливими до графіки застосунками – наприклад, запускають гру у фоні під час монтажу відео або проводять паралельно рендеринг ігор та стрімів. Оскільки сучасні графічні карти мають обмежений обсяг відеопам’яті та ядра для обробки шейдерів, одночасний доступ різних процесів призводить до падіння частоти кадрів, затримок в інтерфейсі та навіть зависань [2].

Конфлікти за дисковий простір – системний диск, на якому розміщуються файли операційної системи та тимчасові файли програм, вмить заповнюється в разі завантаження великих обсягів даних. Наприклад, одночасне завантаження оновлень Windows, відео з інтернету та створення великих резервних копій може призвести до того, що системі не вистачить

місця для створення нових тимчасових файлів, спричиняючи втрачені операції запису і довге очікування підтвердження завершення завдань.

Конфлікти за мережеві ресурси – у мережах зі спільною пропускною здатністю різні програми (хмарні сховища, відеоконференції) конкурують за канали передачі даних [3]. Під час інтенсивного завантаження або відвантаження великих файлів загальна смуга пропускання може бути повністю зайнята, що веде до затримок у роботі інших мережевих сервісів, втрати пакетів і зниження якості зв’язку.

Таким чином, правильний і завчасний розподіл цих ресурсів між процесами є критичним чинником для забезпечення безперебійної та продуктивної роботи системи. Лише за умови ефективного відстеження використання компонентів апаратного забезпечення та вчасного реагування на потенційні перевантаження можна суттєво покращити користувацький досвід, знизити ризик помилок і тривалих простоїв, а також продовжити термін служби обладнання шляхом уникнення екстремальних робочих режимів.

1.2 Порівняльна характеристика існуючих рішень для оптимізації системних ресурсів

Існують декілька схожих застосунків, які частково вирішують проблему конфліктів програмного забезпечення, проте кожен із них має власні обмеження.

Windows Performance Monitor – вбудований у Windows інструмент для збору детальної статистики про використання центрального процесора, оперативної пам’яті, дискового підсистеми та мережевих інтерфейсів у реальному часі з можливістю зберігати історію лічильників у журналах. У цьому застосунку можна налаштувати лічильники з інтервалами збору даних, а також створювати власні шаблони їх збору, які можна автоматизувати через скрипти. Незважаючи на широкі можливості

налаштування, інтерфейс залишається доволі технічним: користувач повинен вручну створювати колекції лічильників, обирати пороги для сповіщень та обробляти зібрані дані самостійно. Унаслідок цього Windows Performance Monitor більше підходить для системних адміністраторі, а не для пересічного користувача, який прагне швидкого та зручного рішення для запобігання конфліктам ресурсів.

Process Lasso – це застосунок для автоматичного керування пріоритетами процесів у Windows, що використовує алгоритм для динамічного регулювання пріоритетів та на основі аналізу навантаження. Завдяки налаштовуваним профілям та правилам користувач може визначити, які програми мають пріоритет, а які – бути обмеженими в ресурсах, а також створити власні сценарії автоматизації. Водночас інтерфейс Process Lasso має надмірну насиченість налаштувань і відсутність зрозумілих підказок, що ускладнює роботу для нових користувачів. Крім того, без коректного налаштування політик пріоритетів може виникнути ситуація, коли важливі процеси отримують занадто низький пріоритет, що призводить до некоректної роботи або уповільнення критичних застосунків.

SysGauge – надає можливості моніторингу в реальному часі з графічною візуалізацією трендів та створенням звітів. Цей застосунок підтримує кастомні сповіщення при перевищенні порогів завантаження ресурсів. Однак цей інструмент більше нагадує покращений варіант штатного моніторингу у системі Windows: він не пропонує жодних механізмів для прогнозування конфліктів або автоматичного регулювання пріоритетів, тому користувачеві доводиться самостійно аналізувати графіки і вчасно реагувати на сповіщення.

htop – інтерактивний процес-менеджер для Unix-подібних систем із підтримкою інтерфейсу ncurses, який показує дерево процесів, завантаження CPU по ядрам, використання пам’яті та swap у реальному часі. Надає гнучкі фільтри, сортування та можливість взаємодії з процесами для зміни пріоритетів прямо з термінала без графічного навантаження. Проте

htop не має графічної оболонки для Windows і вимагає навички роботи з командним рядком, що робить його майже недоступним для звичайних користувачів без досвіду роботи в терміналі. Цей інструмент скоріше підходить для досвідчених адміністраторів Linux/Unix, які цінують швидкість та мінімальне навантаження на систему.

Zabbix – велика комплексна платформа для централізованого моніторингу серверів, мережевого обладнання та віртуальних машин. Пропонує гнучкі шаблони моніторингу, побудову дашбордів, систему тригерів і сповіщень. Підтримує історію метрик і дозволяє візуалізувати тренди. Однак розгортання Zabbix вимагає налаштування серверної інфраструктури, агентів на хостах і навчання роботі з веб-інтерфейсом, що є надмірно складним для одноосібного користувача робочої станції.

Prometheus + Grafana – ця комбінація є поєднанням спеціалізованої СУБД для метрик і потужного візуалізатора. Prometheus збирає часові ряди даних через HTTP, а Grafana відтворює їх у вигляді інтерактивних графіків і панелей. Система дозволяє опитувати метрики в реальному часі, створювати складні запити та будувати узагальнені показники. Проте для настільного комп’ютера цей стек надто громіздкий: він призначений для середовищ DevOps і потребує налаштування експорту метрик, агента Node Exporter і постійно працюючих серверів.

Таким чином, усі розглянуті рішення демонструють сильні сторони в сфері моніторингу та реактивного керування ресурсами, але жодне не пропонує поєднання простого інтерфейсу та можливості проактивного прогнозування конфліктів. Для звичайного користувача, що хоче захистити свій комп’ютер від гальмувань без глибокого занурення в технічні деталі, потрібна система яка могла б об’єднати кращі практики з моніторингу та машинного навчання, надаючи кінцевому користувачеві максимальний комфорт при мінімальних витратах часу на налаштування.

1.3 Обґрунтування актуальності інтелектуальної системи прогнозування конфліктів

Розвиток інформаційних технологій та їх інтеграція у повсякденне життя призвели до ситуації, коли користувач регулярно працює з великою кількістю різних програм на своєму комп'ютері. Це створює передумови для зростання актуальності проблеми конфліктів програмного забезпечення, особливо на комп'ютерах з обмеженими технічними характеристиками.

По-перше, варто звернути увагу на масштаб проблеми конфліктів програмного забезпечення: за результатами експериментів було показано, що через нерівномірний розподіл ресурсів серед віддалених віртуальних машин спостерігається в середньому 5,9 % зниження їх продуктивності – це можна побачити на рисунку 1.1 [4].

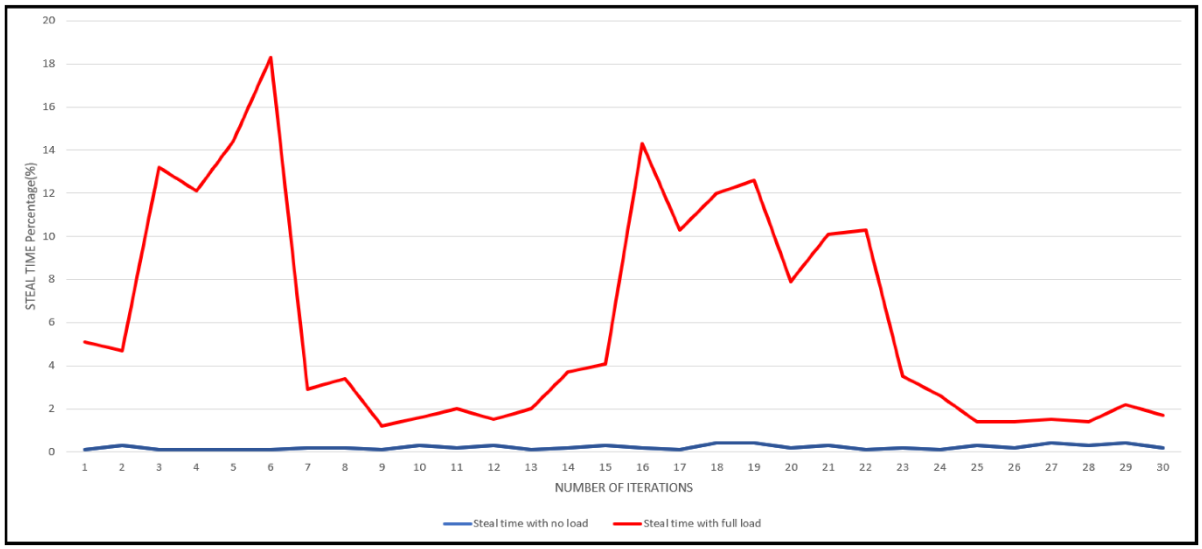


Рисунок 1.1 – Час втрати процесорного часу [4]

Друга причина актуальності розробки системи прогнозування конфліктів полягає у відсутності доступних інструментів для превентивного виявлення потенційних проблем з продуктивністю. Існуючі рішення

здебільшого фокусуються на реактивному підході: вони виявляють та усувають наслідки, але не запобігають самим конфліктам [4]. Реактивне управління ресурсами змушує користувачів витрачати до 20–25 % робочого часу на усунення наслідків [5].

Третім важливим аргументом є зростаючі вимоги сучасного програмного забезпечення до ресурсів. Розмір компонентів програмного забезпечення у хмарному середовищі зростає щороку, а вимоги до оперативної пам’яті та процесорної потужності збільшуються ще швидше.

Четвертім фактором є економічний стан користувачів. Регулярні конфлікти програмного забезпечення іноді сприймаються недосвідченими користувачами як ознака застарілості обладнання, що призводить до передчасної заміни комп’ютерів.

П’ятим аргументом є технологічна готовність. Сучасні інструменти і алгоритми можуть забезпечити точність прогнозування при помірних обчислювальних витратах.

Шостим фактором актуальності є освітній аспект. Інтелектуальна система прогнозування конфліктів може не лише попереджати користувачів про потенційні проблеми, а й пояснювати причини їх виникнення та рекомендовані способи їх вирішення. Це сприятиме підвищенню технічної грамотності користувачів та їхньої здатності ефективно управляти програмним забезпеченням на своїх комп’ютерах. Тим паче, що зараз більшість работодавців очікує, що нові працівники вміють добре користуватися комп’ютером. Тобто у довгостроковій перспективі це може призвести до формування більш відповідального та обізнаного підходу до використання комп’ютерних технологій.

Отже, розробка інтелектуальної системи прогнозування конфліктів програмного забезпечення є актуальним та перспективним напрямком досліджень, що відповідає сучасним тенденціям розвитку інформаційних технологій та потребам користувачів. Реалізація такої системи дозволить підвищити ефективність використання комп’ютерного обладнання, знизити

витрати часу на вирішення технічних проблем та покращити загальний досвід взаємодії користувачів з комп’ютером.

1.4 Аналіз потреб користувачів та вимог до інтерфейсу системи

1.4.1 Сегментація цільової аудиторії системи

Для ефективної розробки та впровадження інтелектуальної системи прогнозування конфліктів програмного забезпечення необхідно провести ретельний аналіз потреб цільової аудиторії та визначити оптимальні підходи до проектування користувацького інтерфейсу. Цей підрозділ присвячений дослідженню особливостей взаємодії користувачів різних рівнів технічної підготовки з системою та формулюванню вимог до її інтерфейсу.

Потенційних користувачів системи прогнозування конфліктів програмного забезпечення можна умовно розподілити на кілька категорій за рівнем технічної підготовки та специфікою використання комп’ютера:

Користувачі базового рівня – користувачі з мінімальним досвідом роботи з комп’ютером, які не розуміють технічних деталей функціонування операційної системи та конкурентного розподілу ресурсів. Для них виникнення проблем із продуктивністю часто викликає розгубленість. Такі користувачі потребують максимально спрощеного інтерфейсу з мінімумом технічних термінів та інтуїтивно зрозумілими рекомендаціями.

Користувачі середнього рівня – регулярно використовують комп’ютер для роботи та розваг, мають базове розуміння принципів функціонування операційної системи, але не володіють глибокими знаннями щодо оптимізації продуктивності. Вони здатні виконувати базові дії для покращення роботи системи за наявності чітких інструкцій та можуть розуміти нескладні технічні пояснення.

Досвідчені користувачі – люди з глибоким розумінням архітектури комп’ютерних систем, які активно використовують спеціалізоване

програмне забезпечення, що потребує значних ресурсів: відеомонтаж, розробка програмного забезпечення тощо. Вони здатні самостійно діагностувати та вирішувати багато проблем із продуктивністю, але потребують детальної технічної інформації та розширених можливостей налаштування системи.

IT-спеціалісти та системні адміністратори – це група професіоналів, які відповідають за підтримку комп'ютерних систем у організаціях. Для них критично важлива можливість централізованого моніторингу, глибокої діагностики проблем та автоматизації процесів оптимізації роботи програмного забезпечення на багатьох робочих станціях одночасно.

1.4.2 Вимоги до інтерфейсу системи

Враховуючи різноманітність цільової аудиторії та множину сценаріїв використання, до інтерфейсу системи прогнозування конфліктів програмного забезпечення висуваються наступні вимоги:

Багаторівневість подання інформації – інтерфейс повинен забезпечувати різні рівні деталізації технічної інформації залежно від потреб та рівня підготовки користувача. Для початківців важливі спрощені індикатори стану системи, тоді як для професіоналів необхідний доступ до детальних метрик та налаштувань.

Інтуїтивність та відсутність перевантаження – базовий інтерфейс має бути зрозумілим без спеціальної підготовки та не містити надмірної кількості елементів керування. Доступ до розширених функцій доцільно реалізувати через додаткові вкладки або режим «експерта».

Проактивність та контекстна допомога – система повинна не лише реагувати на запити користувача, але й самостійно пропонувати релевантну інформацію та рекомендації залежно від поточного стану комп'ютера. Кожен елемент інтерфейсу має супроводжуватися контекстними підказками, що пояснюють його призначення та можливі дії.

Візуалізація даних – використання графіків, діаграм та інших візуальних елементів для наочного представлення стану системи, трендів використання ресурсів та прогнозів потенційних конфліктів. Візуалізація має бути інформативною та не переобтяженою деталями, з можливістю поглиблення аналізу за потреби.

Гнучкість налаштувань – можливість адаптації інтерфейсу до потреб конкретного користувача, включаючи вибір рівня деталізації інформації, частоти сповіщень, кольорових схем тощо.

Мультимодальність взаємодії – підтримка різних способів взаємодії з системою, включаючи графічний інтерфейс, системні сповіщення, спливаючі вікна, інтеграцію з центром сповіщень операційної системи тощо.

Локалізація та доступність – підтримка різних мов інтерфейсу та забезпечення доступності для користувачів з особливими потребами, включаючи адаптацію для програм читання з екрану, налаштування контрастності та розміру елементів інтерфейсу.

1.4.3 Принципи формування рекомендацій користувачам

Особливої уваги заслуговують принципи формування рекомендацій щодо запобігання та вирішення конфліктів програмного забезпечення, оскільки саме цей аспект визначає практичну корисність системи для користувачів різних рівнів підготовки:

Адаптивність складності – система повинна адаптувати рівень технічної деталізації рекомендацій відповідно до профілю користувача. Початківцям надаються прості покрокові інструкції з мінімумом технічних термінів, тоді як для експертів формуються рекомендації з використанням професійної термінології та посиланнями на більш складні інструменти оптимізації.

Освітній компонент – разом із конкретними рекомендаціями бажано надавати стислі пояснення причин виникнення проблеми та механізму дії запропонованого рішення, що сприятиме підвищенню технічної грамотності користувачів.

Моніторинг ефективності – система повинна відстежувати результати впровадження рекомендацій та адаптувати майбутні пропозиції на основі їх ефективності для конкретного користувача та його апаратної конфігурації.

1.4.4 Інтеграція з операційною системою та іншими застосунками

Для забезпечення максимальної ефективності система прогнозування конфліктів програмного забезпечення повинна органічно інтегруватися з операційною системою та іншими застосунками:

Системні сповіщення – використання стандартних механізмів сповіщень операційної системи для інформування користувача про потенційні проблеми.

Інтеграція з планувальником завдань – можливість аналізу запланованих завдань та попередження про потенційні конфлікти, що можуть виникнути під час їх виконання.

Взаємодія з диспетчером завдань – доповнення стандартних інструментів моніторингу системи функціями прогнозування та оптимізації.

Інтеграція з програмами віртуалізації – особлива увага до прогнозування конфліктів при роботі з віртуальними машинами, які особливо інтенсивно використовують системні ресурси.

Взаємодія з хмарними сервісами – можливість синхронізації даних про продуктивність системи із хмарними сервісами для аналізу та порівняння з іншими користувачами.

1.5 Постановка задачі розробки системи прогнозування конфліктів програмного забезпечення

На основі проведеного аналізу предметної галузі, порівняльного огляду існуючих рішень та аналізу актуальності задачі стає очевидним, що створення системи прогнозування конфліктів програмного забезпечення є актуальним та важливим завданням. Основною метою розробки такої системи є створення інтелектуального застосунку, здатного заздалегідь виявляти потенційні конфлікти між програмами за системні ресурси та надавати користувачам проактивні рекомендації щодо їх запобігання.

На відміну від існуючих рішень, які переважно застосовують реактивний підхід, розроблювана система повинна використовувати предиктивні алгоритми для завчасного виявлення потенційних конфліктів. Це дозволить користувачам приймати обґрунтовані рішення щодо запуску програм та планування їх роботи.

Система прогнозування конфліктів програмного забезпечення повинна забезпечувати моніторинг ключових системних ресурсів з можливістю візуалізації даних у режимі реального часу. Важливим аспектом є профілювання програмного забезпечення – збір та аналіз інформації про використання ресурсів різними застосунками в різних режимах роботи для створення їх ресурсних профілів. На основі цих профілів та поточного стану системи має здійснюватися прогнозування потенційних конфліктів при спробі запуску нових застосунків.

Користувачі повинні отримувати зрозумілі сповіщення про можливі конфлікти перед запуском програм.

Для подальшого вдосконалення алгоритмів прогнозування система повинна вести статистику про частоту виникнення конфліктів, їх вплив на продуктивність. Важливим аспектом є можливість адаптації системи до особливостей використання комп’ютера конкретним користувачем з метою підвищення точності прогнозування конфліктів.

З технічної точки зору, система не повинна значно навантажувати процесор та оперативну пам’ять, щоб не створювати додаткових конфліктів за ресурси. Вона має бути сумісною з різними версіями Windows. Алгоритми прогнозування повинні забезпечувати високу точність для найпоширеніших типів конфліктів. Інтерфейс користувача має бути інтуїтивно зрозумілим для користувачів з різним рівнем технічної підготовки.

Архітектура системи повинна бути модульною та включати компоненти для моніторингу ресурсів, аналізу даних, прогнозування конфліктів, взаємодії з користувачем та зберігання даних. Такий підхід забезпечить масштабованість системи та можливість додавання нових функцій без значної перебудови основних компонентів.

Для розробки ефективних алгоритмів прогнозування конфліктів програмного забезпечення будуть використані алгоритми машинного навчання. Це дозволить досягти високої точності прогнозування різних типів конфліктів за різних умов використання комп’ютера.

В результаті успішної реалізації поставлених завдань буде створено інноваційну систему, яка забезпечить підвищення стабільності роботи комп’ютера, скорочення часу на вирішення технічних проблем, оптимізацію використання наявних системних ресурсів та підвищення технічної грамотності користувачів. Важливим результатом також стане збір цінних статистичних даних про особливості використання комп’ютерів, що може бути використано для подальших досліджень у галузі оптимізації роботи програмного забезпечення.

Таким чином, розробка системи прогнозування конфліктів програмного забезпечення є завданням, яке відповідає сучасним тенденціям розвитку інформаційних технологій та потребам користувачів. Реалізація такої системи дозволить значно покращити досвід використання комп’ютера для багатьох користувачів та, особливо для тих, хто використовує комп’ютери з обмеженими технічними характеристиками.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Огляд наукових джерел та сучасних підходів

Проблема прогнозування та попередження конфліктів програмного забезпечення у комп’ютерних системах є складною задачею, що поєднує методи системного аналізу, машинного навчання, операційних систем та взаємодії людини з комп’ютером. Кожен з цих методів має свої переваги та обмеження.

2.1.1 Статистичні методи моніторингу та прогнозування

Перший напрямок досліджень базується на використанні статистичних методів для аналізу історичних даних про використання системних ресурсів. Ці підходи ґрунтуються на припущенні, що поведінка програмного забезпечення має певні закономірності, які можна виявити та використати для прогнозування майбутніх станів системи.

Класичні моделі ARIMA і SARIMA добре виявлятимуть сезонні коливання навантаження на процесор і пам’ять, але їх точність помітно впаде при раптових стрибках активності. Поєднання цих моделей із рекурентними нейронними мережами дозволяє знизити похибки короткострокового прогнозу CPU майже вдвічі [13].

Багатовимірні статистичні моделі дозволяють аналізувати взаємозв’язки між різними системними ресурсами. Використання кореляційного аналізу та регресійних моделей дає можливість виявити, як зміна одного параметра впливає на інші. Це особливо важливо для розуміння каскадних ефектів у системі, коли перевантаження процесора призводить до збільшення використання оперативної пам’яті через кешування, що, у свою чергу, може спричинити активізацію файлу підкачки.

Експоненціальне згладжування та його варіації використовуються для виявлення трендів у використанні ресурсів та прогнозування короткострокових змін. Ці методи особливо ефективні для систем із відносно стабільною поведінкою, де можна виділити базовий рівень навантаження та відхилення від нього.

Однак статистичні методи мають суттєві обмеження при роботі з нелінійними залежностями та раптовими змінами в поведінці системи. Запуск нової ресурсоємної програми може кардинально змінити характер використання ресурсів, що робить попередні статистичні моделі неточними.

2.1.2 Підходи на основі порогових значень та правил

Другий популярний напрямок базується на використанні експертних знань для створення систем правил та порогових значень. Ці підходи спираються на досвід системних адміністраторів та розробників у визначенні критичних рівнів використання ресурсів.

Фіксовані порогові значення (наприклад, 85 % CPU або 90 % RAM) часто генерують численні хибні тривоги, тоді як динамічне коригування меж на основі історичної статистики та допустимого рівня аномалій значно скорочує кількість непотрібних спрацювань [14].

Адаптивні порогові системи намагаються вирішити цю проблему шляхом динамічного коригування порогових значень на основі історичних даних [15]. Наприклад, якщо система регулярно працює з високим навантаженням на процесор без виникнення проблем, пороги можуть бути підвищені.

Системи на основі правил дозволяють створювати більш складні логічні конструкції для виявлення потенційних конфліктів. Наприклад, правило може визначати, що одночасний запуск двох відеоредакторів на системі з менш ніж 16 ГБ оперативної пам’яті є ризикованим. Такі системи

можуть враховувати комбінації різних факторів та створювати більш точні прогнози.

Експертні системи розширюють підхід на основі правил, включаючи механізми логічного виведення та базу знань. Вони можуть не лише виявляти потенційні проблеми, але й пропонувати рішення на основі накопиченого досвіду.

Основною перевагою цих підходів є їх прозорість та можливість пояснення прийнятих рішень. Користувач може зрозуміти, чому система генерує певне попередження. Однак створення та підтримка комплексних систем правил вимагає значних зусиль експертів, а самі правила можуть бути занадто жорсткими для складних сценаріїв використання.

2.1.3 Методи машинного навчання для аналізу поведінки системи

Класифікаційні алгоритми – SVM, випадкові ліси, градієнтний бустинг – успішно відокремлюють стабільні стани від конфліктних [16].

Регресійні методи займають особливе місце в прогнозуванні споживання ресурсів програмним забезпеченням. Лінійна регресія та її розширення дозволяють моделювати залежність між характеристиками програм та їх ресурсними потребами. Random Forest Regressor демонструє високу ефективність у багатьох задачах прогнозування завдяки його здатності ураховувати складні нелінійні залежності між різними характеристиками даних. Регресійні моделі на основі градієнтного бустингу, наприклад XGBoost та LightGBM, успішно показують себе при роботі з великими наборами данних. Полиноміальна регресія ефективна для моделювання складних залежностей між розміром файлів програм, кількістю одночасно відкритих вікон та споживанням ресурсів. Регресійні методи також дозволяють кількісно оцінити вплив окремих характеристик програм на споживання ресурсів, що важливо для розуміння причин конфліктів.

Нейронні мережі, особливо рекурентні архітектури, такі як LSTM (Long Short-Term Memory), показують хороші результати в прогнозуванні часових рядів системних метрик [17]. Вони здатні виявляти складні нелінійні залежності та довгострокові тренди в даних, що робить їх особливо корисними для систем з нерегулярною поведінкою.

Методи навчання без учителя, включаючи кластеризацію та виявлення аномалій, дозволяють автоматично ідентифікувати незвичайні паттерни в роботі системи. А алгоритми Isolation Forest та One-Class SVM, здатні помічати різні аномальні стани навіть якщо вони не мають знань про те, як ці стани виглядають.

Ансамблеві методи поєднують кілька різних алгоритмів для підвищення точності та надійності прогнозів. Наприклад, можна комбінувати статистичні моделі для виявлення трендів із алгоритмами машинного навчання для класифікації аномалій.

Алгоритми глибокого навчання, включаючи автоенкодери та генеративні моделі, використовуються для створення детальних моделей нормальної поведінки системи. Відхилення від цих моделей можуть вказувати на потенційні проблеми.

2.1.4 Підходи на основі моделювання та симуляції

Четвертий напрямок фокусується на створенні математичних моделей поведінки системи та використанні симуляції для прогнозування наслідків запуску нових програм або зміни конфігурації системи.

Теорія черг ефективно прогнозує час очікування та пропускну здатність процесів, а дискретно-подійні симуляції забезпечують детальний рівень подій (запуск, запит і звільнення ресурсів) для точного виявлення вузьких місць у системі [18].

Агентні моделі представляють кожну програму як окремий агент із власними потребами в ресурсах та поведінкою. Взаємодія між агентами

симулюється для прогнозування загального стану системи. Такі моделі особливо корисні для розуміння складних сценаріїв із багатьма одночасно працюючими програмами.

Моделі на основі теорії ігор розглядають конкуренцію за ресурси як стратегічну взаємодію між програмами. Кожна програма намагається максимізувати свою продуктивність, що може призводити до конфліктів. Ці моделі дозволяють знаходити оптимальні стратегії розподілу ресурсів.

Дискретно-подійні симуляції моделюють роботу системи як послідовність подій, таких як запуск програм, запити на ресурси та їх звільнення. Такі моделі дозволяють детально проаналізувати динаміку системи та виявити вузькі місця.

Сучасні дослідження все частіше фокусуються на поєднанні різних методів для створення більш надійних та точних систем прогнозування.

Багаторівневі архітектури поєднують різні методи на різних рівнях абстракції. Наприклад, статистичні методи можуть використовуватися для виявлення загальних трендів, алгоритми машинного навчання – для класифікації аномалій, а системи правил – для генерації конкретних рекомендацій.

Адаптивні системи автоматично вибирають найкращий метод прогнозування залежно від поточного стану системи та доступних даних. Наприклад, для систем із стабільною поведінкою можуть використовуватися статистичні методи, а для систем із нерегулярною активністю – алгоритми машинного навчання.

Системи реального часу поєднують швидкі евристичні методи для негайного реагування на критичні ситуації з більш складними алгоритмами для довгострокового планування та оптимізації.

Інтеграція з операційною системою дозволяє створювати системи, що використовують внутрішні механізми комп’ютера для моніторингу та управління ресурсами. Такі системи можуть мати доступ до детальної

інформації про стан процесів та використовувати вбудовані засоби оптимізації.

Ансамблеві методи становлять окрему категорію гібридних підходів, де декілька незалежних моделей працюють паралельно, а їхні результати комбінуються для отримання більш точного прогнозу. Такий підхід дозволяє компенсувати слабкості окремих алгоритмів та підвищити загальну надійність системи. Методи голосування, зважування та стекінгу використовуються для агрегації результатів різних моделей.

Мета-навчання представляє сучасний напрямок розвитку адаптивних систем, де алгоритми не просто вибирають готові методи, а навчаються створювати нові підходи на основі аналізу попереднього досвіду. Такі системи здатні швидко адаптуватися до нових умов та автоматично налаштовувати свої параметри без втручання людини.

2.1.5 Методи профілювання та характеризації програмного забезпечення

Окремим важливим напрямком є розробка методів для створення детальних профілів поведінки різних типів програмного забезпечення.

Статичний аналіз коду дає змогу оцінити потенційне споживання ресурсів ще до запуску програми, а легковісна інструментація в реальному часі збирає точні метрики з мінімальним впливом на продуктивність системи [19].

Динамічне профілювання передбачає моніторинг реального використання ресурсів програмами під час їх роботи. Це дозволяє створювати точні профіли для різних режимів роботи програм та різних типів даних, що обробляються.

Бенчмаркінг та тестування продуктивності використовуються для систематичного вивчення поведінки програм у контрольованих умовах. Це

дозволяє створювати стандартизовані профіли програм та порівнювати різні версії або конфігурації.

Кластеризація програм за їх ресурсними характеристиками дозволяє групувати схожі за поведінкою програми та створювати узагальнені моделі для кожної групи. Це спрощує прогнозування поведінки нових програм на основі їх подібності до відомих категорій.

Таким чином, сучасні підходи до прогнозування конфліктів програмного забезпечення охоплюють широкий спектр методів від простих статистичних моделей до складних систем машинного навчання. Кожен підхід має свої переваги та обмеження, що обумовлює необхідність їх критичного порівняння та вибору оптимального рішення для конкретних умов застосування.

2.1.6 Гібридні підходи

Розробка гібридних систем може поєднувати регресійне прогнозування споживання ресурсів з подальшою класифікацією потенційних конфліктних ситуацій. Такий двоетапний підхід дозволяє максимально використовувати переваги різних алгоритмів машинного навчання.

Наприклад, на першому етапі регресійна модель Random Forest Regressor прогнозує кількісні показники споживання ресурсів програмами на основі їх характеристик. Ці моделі можуть враховувати розмір виконуваних файлів, тип програми, кількість одночасно відкритих вікон, історичні дані про попереднє використання та інші релевантні ознаки.

На другому етапі класифікаційні алгоритми або системи правил аналізують прогнозовані значення споживання ресурсів у контексті загальної доступності системних ресурсів. Це дозволяє виявляти ситуації, коли сумарне навантаження може перевищити можливості системи та призвести до конфліктів.

Такий підхід вирішує проблему холодного старту для нових застосунків, коли відсутні історичні дані про їх поведінку. Регресійні моделі можуть прогнозувати споживання ресурсів на основі статичних характеристик програм, що дозволяє системі надавати обґрунтовані рекомендації навіть для раніше невідомих застосунків.

Гібридні системи також забезпечують кращу інтерпретованість результатів порівняно з повністю чорноскриньковими підходами. Користувачі можуть бачити як прогнозовані значення споживання ресурсів, так і логіку прийняття рішень про потенційні конфлікти.

2.2 Класифікація та порівняння методів

Для ефективного аналізу існуючих підходів до прогнозування конфліктів програмного забезпечення доцільно провести їх систематизацію за кількома ключовими критеріями. Така класифікація дозволить краще зрозуміти переваги та недоліки різних методів, а також визначити оптимальні підходи для розробки інтелектуальної системи прогнозування.

2.2.1 Класифікація за типом алгоритму

За типом алгоритму методи прогнозування конфліктів програмного забезпечення можна розділити на п’ять основних категорій: детерміністичні статистичні та евристичні методи, алгоритми машинного навчання з учителем та без, а також імітаційні та модельні підходи.

Детерміністичні статистичні методи базуються на математичному аналізі історичних даних та виявленні закономірностей у часових рядах. До цієї категорії належать моделі ARIMA, експоненціальне згладжування, регресійний аналіз та кореляційні методи. Ці підходи передбачають, що майбутня поведінка системи може бути передбачена на основі минулих спостережень із використанням детермінованих математичних формул.

Евристичні методи на основі правил використовують експертні знання та досвід для створення логічних конструкцій, що описують умови виникнення конфліктів. Сюди входять прості порогові системи, адаптивні пороги, продукційні системи правил та експертні системи. Ці методи спираються на явно сформульовані знання предметної області.

Алгоритми машинного навчання з учителем навчаються на попередньо розмічених даних для класифікації станів системи або прогнозування значень метрик. До цієї групи належать Support Vector Machines, випадкові ліси, градієнтний бустинг, нейронні мережі прямого поширення та рекурентні архітектури, такі як LSTM.

Методи навчання без учителя автоматично виявляють приховані закономірності в даних без попередньої розмітки. Головними методами є алгоритми групування даних Isolation Forest та One-Class SVM.

Імітаційні та модельні підходи створюють математичні або комп’ютерні моделі поведінки системи для прогнозування наслідків різних сценаріїв. Включають моделі черг, агентні моделі, теоретико-ігрові підходи та дискретно-подійні симуляції.

2.2.2 Класифікація за типом задачі

Важливим критерієм класифікації є тип задачі машинного навчання, яку вирішує метод. За цим критерієм можна виділити три основні категорії: класифікаційні та регресійні методи, а також комбіновані підходи.

Класифікаційні методи зосереджуються на виявленні конфліктних ситуацій шляхом віднесення поточного стану системи до одного з класів: стабільний, потенційно конфліктний або критичний. До цієї категорії належать SVM, випадкові ліси для класифікації, нейронні мережі з сигмоїдною або softmax активацією, а також алгоритми виявлення аномалій. Ці методи зазвичай використовують дискретні мітки класів та оцінюються метриками точності та повноти.

Регресійні методи фокусуються на прогнозуванні кількісних показників споживання ресурсів програмами. Вони включають Random Forest Regressor, лінійну регресію та XGBoost. Ці методи прогнозують неперервні значення, такі як обсяг споживаної оперативної пам’яті, процесорного часу або дискового простору.

Комбіновані підходи поєднують регресійні та класифікаційні компоненти для створення більш комплексних рішень. Наприклад, система може спочатку прогнозувати споживання ресурсів за допомогою регресійних моделей, а потім використовувати ці прогнози для класифікації рівня ризику конфліктів. Такі гібридні системи дозволяють використовувати переваги обох підходів та забезпечують як кількісні прогнози, так і категоричні оцінки ризику.

2.2.3 Класифікація за горизонтом прогнозування

Короткострокові методи орієнтовані на миттєве виявлення критичних ситуацій і формування оперативних рішень. Вони базуються на простих евристичних правилах або швидких статистичних оцінках, що дозволяє системі реагувати з мінімальною затримкою.

Середньострокові методи спрямовані на прогнозування розвитку ситуації в межах найближчого робочого періоду користувача. Тут застосовують трохи складніші алгоритми – легкі моделі машинного навчання та аналіз трендів, які забезпечують баланс між швидкістю обробки і точністю прогнозів.

Довгострокові методи зосереджені на виявленні загальних паттернів використання системи та плануванні ресурсів на майбутні періоди. У цьому випадку часто використовують ресурсоємні алгоритми глибокого навчання або симуляційні моделі для оптимізації розподілу ресурсів і мінімізації ризиків.

2.2.4 Класифікація за типом ресурсів

За типом ресурсів, конфлікти за які прогнозуються, методи поділяються на два типи.

Універсальні методи здатні аналізувати конфлікти за всіма типами системних ресурсів одночасно, використовуючи багатовимірні моделі та комплексні показники системного навантаження.

Спеціалізовані методи фокусуються на конкретних типах ресурсів: процесорному часі, оперативній пам’яті, дисковому простору, мережевій пропускній здатності або графічних ресурсах. Така спеціалізація дозволяє досягти вищої точності для конкретного типу конфліктів.

2.2.5 Класифікація за рівнем складності реалізації

Прості методи легко реалізувати та налаштувати, вони потребують мінімальних обчислювальних ресурсів та можуть працювати на системах із обмеженими характеристиками. Включають порогові системи та базові статистичні підходи.

Середньої складності методи вимагають більше ресурсів для навчання та роботи, але забезпечують кращу точність. Сюди входять класичні алгоритми машинного навчання та статистичні моделі часових рядів.

Складні методи потребують значних обчислювальних ресурсів та експертизи для налаштування, але можуть забезпечити найвищу точність. Включають глибокі нейронні мережі, ансамблеві методи та складні симуляційні моделі.

2.2.6 Точність прогнозування

Точність прогнозування є одним із найважливіших критеріїв оцінки ефективності методів.

Емпіричні дослідження показують різну ефективність методів залежно від типу системи та характеру навантаження.

Статистичні методи, такі як моделі ARIMA, демонструють високу точність у середовищах із регулярною поведінкою та чіткими трендами. Наприклад, у дослідженні зазначено, що модель SARIMA ефективно прогнозує використання CPU в кластерах із вираженою сезонністю, забезпечуючи низьку MAE. Однак у випадках із нерегулярним або нестабільним навантаженням, де відсутні чіткі сезонні патерни, точність прогнозування значно знижується, що свідчить про обмеження ARIMA-моделей у таких умовах.

Дослідження показують, що методи на основі правил можуть досягати високої точності в добре вивчених сценаріях, однак їх ефективність значно знижується при застосуванні до нових ситуацій. Прості порогові системи виявлення аномалій можуть мати високий рівень хибних спрацьовувань у динамічних середовищах.

Застосування алгоритмів машинного навчання, таких як випадкові ліси та градієнтний бустинг, дозволяє досягати високої точності у прогнозуванні складних багатовимірних задач. Нейронні мережі типу LSTM показують високу точність при аналізі часових рядів зі складними паттернами.

Методи виявлення аномалій можуть досягати точності, однак у динамічних середовищах часто спостерігається високий рівень хибних спрацьовувань.

Імітаційні моделі можуть досягати високої точності у специфічних сценаріях, однак їх створення та калібрування є трудомістким процесом.

2.2.7 Швидкість роботи та обчислювальна складність

Швидкість роботи відіграє вирішальну роль у системах реального часу, особливо коли йдеться про взаємодію з користувачем. Статистичні

підходи характеризуються мінімальними обчислювальними витратами та здатні видавати прогнози за долі секунди. Наприклад, метод експоненціального згладжування після початкової налаштування здатен миттєво оновлювати значення без додаткових обчислень.

Системи на основі правил працюють з надзвичайною швидкістю: навіть досить складні експертні механізми здатні робити логічні висновки за кілька мілісекунд. Алгоритми машинного навчання демонструють різну продуктивність залежно від своєї природи. Найпростішим моделям достатньо обробки самої кількості вхідних ознак, і вони відпрацьовують неймовірно швидко. Комплекти методів на кшталт лісових ансамблів потребують значного часу для навчання й трохи більше зусиль під час прогнозування, оскільки кожне дерево вимагає окремої обробки. Глибокі нейронні мережі можуть заявити про себе вже за десятки мілісекунд або ж займати секунди й навіть більше, коли справа доходить до складних архітектур.

Методи виявлення аномалій також демонструють різні швидкісні характеристики: від відносно швидких підходів, що укладаються в частки секунди, до тих, які зростають у вимогливості до ресурсів з ростом обсягу даних. Найбільш ресурсомісткими вважаються симуляційні моделі: через високу деталізацію та довгі горизонти прогнозування їхні розрахунки можуть тривати від кількох секунд до декількох годин.

2.2.8 Вимоги до даних та навчання

Різні методи мають кардинально різні вимоги до обсягу та якості навчальних даних.

Статистичні методи потребують відносно невеликих обсягів даних - від кількох сотень до кількох тисяч спостережень. Вимагають стаціонарності часових рядів або можливості їх перетворення до стаціонарного вигляду.

Системи правил не потребують навчальних даних в традиційному розумінні, але вимагають глибоких експертних знань для створення якісної бази правил. Процес створення може займати місяці роботи експертів.

Алгоритми машинного навчання потребують значних обсягів розмічених даних: від тисяч до мільйонів прикладів залежно від складності задачі. Глибокі нейронні мережі можуть вимагати десятки мільйонів параметрів для навчання.

Методи без учителя не потребують розмічених даних, але вимагають репрезентативних вибірок нормальної поведінки системи для побудови базової моделі.

Симуляційні моделі потребують детальних знань про архітектуру системи та параметри компонентів, що може бути складно отримати для загальних користувальницьких систем.

2.2.8 Адаптивність та здатність до узагальнення

Адаптація методів до нових умов і здатність узагальнювати знання є ключовими для їх ефективного застосування в реальних сценаріях. Статистичні методи, завдяки онлайн-навчанню та адаптивному згладжуванню, добре справляються з поступовими змінами в поведінці системи, однак стають неефективними при раптових або кардинальних змінах структури даних. Це обмеження може призвести до значних похибок у передбаченнях або рішеннях, якщо своєчасно не виявити зміну контексту.

Системи правил, незважаючи на свою простоту та високу інтерпретованість, практично не здатні самостійно адаптуватися. Кожна зміна умов потребує втручання людини для оновлення логіки. Це ускладнює їх застосування в динамічних або складних середовищах, де варіативність поведінки системи висока.

Алгоритми машинного навчання, особливо з використанням регуляризації та перехресної валідації, мають високу здатність до

узагальнення. Вони ефективно вловлюють приховані патерни у великих наборах даних і можуть адаптуватися до нових ситуацій шляхом дообучення, часто без потреби в кардинальних змінах моделі. Це робить їх одним з найперспективніших напрямів для побудови адаптивних систем.

Методи виявлення аномалій добре формують уявлення про нормальну поведінку системи та виявляють відхилення, проте їхня ефективність знижується при еволюції цієї норми. Для підтримання актуальності моделей потрібне періодичне перенавчання або застосування адаптивних стратегій оновлення.

Симуляційні моделі є найменш гнучкими серед розглянутих підходів. Вони потребують ручного налаштування при будь-яких змінах конфігурації системи, що суттєво обмежує їхнє застосування у змінних або нестабільних середовищах. Проте вони залишаються корисними в контрольованих умовах, де можливі зміни можна заздалегідь передбачити й врахувати.

2.2.9 Аналіз придатності методів для різних сфер застосування

Для персональних комп’ютерів ключовими вимогами є простота використання, мінімальне навантаження на систему та зрозумілість рекомендацій для звичайних користувачів.

Оптимальні методи: комбінація простих порогових систем для миттєвого реагування на критичні ситуації з легкими алгоритмами машинного навчання, такі як випадкові ліси та лінійні моделі, для більш точного прогнозування. Статистичні методи підходять для виявлення регулярних паттернів використання.

Менш придатні методи: складні глибокі нейронні мережі через високе споживання ресурсів, симуляційні моделі через складність налаштування, експертні системи через потребу в спеціалізованих знаннях.

Системи, орієнтовані на ігри та роботу з мультимедіа, характеризуються нерегулярним, але інтенсивним навантаженням на графічні та обчислювальні ресурси.

Оптимальні методи: методи виявлення аномалій для розпізнавання нестандартних навантажень, спеціалізовані алгоритми машинного навчання, налаштовані на графічні та мультимедійні застосунки. Важлива швидкість реакції для уникнення впливу на ігровий досвід.

Менш придатні методи: статистичні методи через нерегулярність навантаження, прості порогові системи через різноманітність сценаріїв використання.

Професійні робочі станції використовуються для ресурсоємних завдань, таких як відеомонтаж, 3D-моделювання, наукові розрахунки, розробка програмного забезпечення.

Оптимальні методи: складні алгоритми машинного навчання, здатні аналізувати багатовимірні залежності між ресурсами. Симуляційні моделі для планування складних робочих процесів. Адаптивні системи правил, налаштовані експертами предметної області.

Менш придатні методи: прості порогові системи через складність робочих процесів, методи з низькою точністю через критичність продуктивності для професійних завдань.

Серверні системи вимагають високої надійності, можливості централізованого управління та інтеграції з існуючими системами моніторингу.

Оптимальні методи: комплексні системи на основі машинного навчання з можливістю обробки великих обсягів даних, інтеграція з корпоративними системами моніторингу, використання симуляційних моделей для планування ємності.

Менш придатні методи: прості локальні рішення без можливості централізованого управління, методи з високими вимогами до інтерактивної взаємодії з користувачем.

2.2.10 Синтез оптимального підходу

На основі проведеного аналізу можна зробити висновок, що оптимальне рішення для системи прогнозування конфліктів програмного забезпечення повинно поєднувати переваги різних підходів у багаторівневій архітектурі.

Перший рівень системи являє собою реактивний компонент, який виконує функції негайного реагування на критичні ситуації. Цей рівень базується на швидких евристичних методах та простих порогових системах, що дозволяє миттєво ідентифікувати очевидні конфлікти та запобігати їх розвитку. Реактивний компонент характеризується мінімальними вимогами до обчислювальних ресурсів та забезпечує базовий захист системи. Його алгоритми оптимізовані для роботи з мінімальною затримкою, що критично важливо для підтримання стабільності системи в умовах високого навантаження.

Другий рівень представляє проактивний компонент, який становить основу функціональності системи прогнозування. Цей рівень використовує алгоритми машинного навчання середньої складності, зокрема випадкові ліси та градієнтний бустинг, для прогнозування потенційних конфліктів на часовому горизонті від декількох хвилин до кількох годин. Проактивний компонент здійснює континуальний аналіз системних параметрів, поведінки користувачів та патернів використання ресурсів. Його моделі навчаються на історичних даних та адаптуються до змін у системному середовищі, забезпечуючи високу точність прогнозування при помірному споживанні обчислювальних ресурсів.

Третій рівень являє собою стратегічний компонент, призначений для довгострокового планування та глибокої оптимізації системи. Цей рівень використовує найсучасніші методи глибокого навчання, нейронні мережі високої складності та симуляційні моделі для аналізу складних залежностей та прогнозування системної поведінки на тривалі періоди. Стратегічний

компонент працює у фоновому режимі, не впливаючи на поточну продуктивність системи, та забезпечує довгострокову оптимізацію архітектури програмного забезпечення. Його алгоритми здатні моделювати складні сценарії взаємодії компонентів та прогнозувати еволюцію системних конфліктів.

Ключовим елементом архітектури є адаптивний компонент, який здійснює інтелектуальний вибір найоптимальніших методів прогнозування залежно від поточних характеристик системи, індивідуальної поведінки користувачів та доступних обчислювальних ресурсів. Цей компонент використовує методи мета-навчання для автоматичної настройки параметрів системи та динамічного перерозподілу обчислювального навантаження між різними рівнями архітектури. Адаптивність забезпечує оптимальний баланс між точністю прогнозування та продуктивністю системи в різноманітних умовах експлуатації.

Інтеграційний рівень архітектури забезпечує взаємодію між компонентами та координацію їх роботи. Цей рівень включає механізми синхронізації, обміну даними та консолідації результатів прогнозування від різних алгоритмів. Інтеграційні компоненти також відповідають за моніторинг ефективності системи та автоматичне налаштування параметрів для досягнення оптимальної продуктивності.

Така багаторівнева архітектура забезпечує створення універсальної системи, придатної для широкого спектру застосувань. Система може ефективно функціонувати як на домашніх комп’ютерах з обмеженими ресурсами, так і на потужних професійних робочих станціях та серверних кластерах. Масштабованість архітектури дозволяє динамічно адаптувати функціональність системи до наявних обчислювальних можливостей та специфічних вимог конкретного застосування.

Запропонований підхід гарантує високу точність прогнозування завдяки використанню ансамблю різних алгоритмів та методів на різних рівнях системи. Водночас, ієрархічна структура забезпечує мінімальний

вплив на продуктивність основних системних процесів, оскільки найбільш ресурсомісткі операції виконуються на фоновому рівні. Система також характеризується високою надійністю завдяки резервуванню функціональності на різних рівнях та можливості деградації до простіших методів у випадку збоїв складних компонентів.

2.3 Аналіз переваг та недоліків існуючих підходів

Усі розглянуті в попередньому розділі методи машинного навчання по-різному поєднують у собі баланс між точністю прогнозування, швидкістю обчислень і прозорістю інтерпретації результатів. Ансамблеві алгоритми, зокрема Random Forest і FastForest, демонструють високу точність за рахунок усереднення результатів багатьох дерев рішень. Водночас за складних або нерівномірно розподілених даних Random Forest може потребувати значних обчислювальних ресурсів: час навчання зростає нелінійно із загальною глибиною дерев і кількістю ознак, що іноді робить його непридатним для режиму реального часу. Особливо це проявляється при роботі з великими наборами даних, де навіть незначне збільшення кількості спостережень призводить до експоненційного зростання обчислювальної складності. FastForest частково вирішує це, скорочуючи вибіркові підгрупи даних і використовуючи оптимізовані процедури обрізки, але за це доводиться платити дещо зниженою стійкістю до складних багатокласових сценаріїв. Крім того, методи випадкового вибіркування у FastForest можуть призводити до втрати важливих патернів у даних, особливо якщо критичні ознаки зустрічаються рідко або мають нерівномірний розподіл у навчальній вибірці.

Isolation Forest, покликаний виявляти аномалії, відзначається лінійною часовою складністю і мінімальними вимогами до пам’яті, однак його архітектура слабша для класичних завдань класифікації: відсутність маркованого навчання позначається на середній точності, а чутливість до

налаштування гіперпараметрів може призводити до пропуску дрібних, але критичних відхилень у даних. Варто зазначити, що ефективність Isolation Forest суттєво залежить від якості попередньої обробки даних та вибору відповідних ознак. У контексті виявлення аномалій у системних ресурсах алгоритм може давати хибнопозитивні результати при роботі з циклічними процесами або сезонними коливаннями навантаження, що вимагає додаткового впровадження механізмів фільтрації та валідації виявлених відхилень.

Градієнтні методи бустингу, такі як XGBoost та AdaBoost, забезпечують одну з найвищих швидкостей навчання й точностей серед ансамблів, оскільки застосовують поступову компенсацію помилок попередніх ітерацій з обов’язковою регуляризацією. Механізм покрокового навчання дозволяє цим алгоритмам адаптуватися до найскладніших закономірностей у даних, водночас зберігаючи відносно невисоку обчислювальну складність порівняно з глибокими нейронними мережами. Однак без автоматизованих механізмів підбору гіперпараметрів ці алгоритми стають складними у впровадженні: від оптимального заданого темпу навчання до кількості слабких учасників моделі залежить не лише точність, а й ризик перенавчання. Особливу увагу слід приділити параметрам регуляризації, оскільки неправильне їх налаштування може призвести до недонавчання моделі або, навпаки, до надмірної складності, що погіршує генералізацію на нових даних.

Лінійні моделі та імовірнісні класифікатори – Logistic Regression, Support Vector Machine та Naive Bayes – відзначаються швидким часом навчання та простою інтерпретацією, що робить їх придатними для обмежених середовищ і стартових прототипів. Їхня математична простота забезпечує стабільність роботи навіть за умов обмежених обчислювальних ресурсів та дозволяє легко інтерпретувати вплив окремих ознак на кінцевий результат. Водночас вони суттєво обмежені у здатності моделювати складні, нелінійні взаємозв’язки у багатовимірних даних, що в реальних

умовах часто призводить до заниженої точності. Зокрема, Support Vector Machine з лінійним ядром може не впоратися з завданнями, де межі між класами мають складну геометричну форму, а Naive Bayes припускає умовну незалежність ознак, що рідко виконується на практиці при аналізі системних показників.

Алгоритми на основі відстані, зокрема K-Nearest Neighbors, прості в реалізації та не потребують стадії навчання, але коштують дорогою обчислювальною складністю під час кожного прогнозу та чутливі до масштабування ознак і вибору метрики. Основною перевагою цього алгоритму є здатність автоматично адаптуватися до локальних особливостей розподілу даних без явного навчання параметричної моделі. Проте цей підхід стає практично неприйнятним для великих наборів даних, оскільки кожне прогнозування вимагає обчислення відстаней до всіх навчальних зразків. Крім того, ефективність алгоритму критично залежить від вибору оптимального значення та метрики відстані, що може потребувати складних процедур крос-валідації для різних типів даних.

Деревоподібні методи, такі як окремі Decision Trees, добре інтерпретуються і підтримують одночасну роботу з числовими й категоріальними атрибутами, проте без регуляризації вони легко перенавчаються й утворюють нестабільні структури за невеликих змін у даних. Перевагою дерев рішень є можливість природного відбору найважливіших ознак та створення зрозумілих правил класифікації, які можуть бути легко інтерпретовані предметними експертами. Однак схильність до перенавчання робить окремі дерева непридатними для складних завдань без застосування методів обрізки або ансамблевих підходів. Крім того, дерева рішень можуть створювати незбалансовані структури, що призводить до упередженості в прогнозах для менш представлених класів.

Нарешті, методи глибокого навчання – штучні нейронні мережі та конволюційні архітектури – мають безпрецедентну здатність урахувати

складні нелінійні патерни та автоматично витягувати ознаки, проте вимагають суттєвих обчислювальних потужностей і великих обсягів даних для тренування. Глибокі мережі особливо ефективні при роботі з високорозмірними даними та можуть виявляти приховані закономірності, недоступні традиційним алгоритмам машинного навчання. Крім того, вони практично непрозорі: пояснення зразків поведінки таких моделей можливе лише через зовнішні фреймворки, що додає ще один рівень складності у розробці та впровадженні. Варто також зазначити, що глибокі мережі схильні до катастрофічного забування при дотренуванні на нових даних та можуть виявляти нестабільність при незначних змінах у вхідних параметрах, що критично важливо для систем реального часу.

Додатковим фактором, який слід враховувати при виборі методу, є здатність алгоритмів працювати з неповними або зашумленими даними. У реальних умовах моніторингу систем часто трапляються ситуації з відсутніми вимірюваннями або некоректними показниками, що може суттєво вплинути на якість прогнозування. Ансамблеві методи, зокрема Random Forest, демонструють природну стійкість до таких проблем.

Таким чином, вибір оптимального методу потребує компромісу між продуктивністю в реальному часі й максимальною точністю розпізнавання, а також урахування можливості інтеграції вбудованих механізмів інтерпретації та обмежень ресурсів цільової платформи. Важливо також враховувати специфіку предметної області та характеристики доступних даних для навчання.

2.4 Висновки щодо доцільності використання методів для вирішення задачі

Порівняльний аналіз методів інтелектуального прогнозування конфліктів програмного забезпечення свідчить про доцільність застосування двоетапного підходу, який поєднує регресійне прогнозування

споживання ресурсів із подальшою класифікацією потенційних конфліктних ситуацій. Така архітектура дозволяє максимально використовувати сильні сторони різних алгоритмів і водночас мінімізувати їхні недоліки завдяки чіткому розмежуванню завдань.

На першому етапі система зосереджується на прогнозуванні обсягу спожитої оперативної пам’яті з використанням Random Forest Regressor. Цей ансамблевий метод демонструє високу точність у випадках нелінійних залежностей між характеристиками застосунків, стійко працює з викидами та пропущеними значеннями, дозволяє інтерпретувати важливість окремих ознак та ефективно функціонує навіть при обмеженому обсязі навчальних даних. Додатковою перевагою є можливість адаптації моделі до змін у поведінці застосунків через регулярне перенавчання.

Другий етап передбачає побудову системи правил на основі прогнозованих значень споживання ресурсів. Порогові рівні та аналіз сумарного навантаження на систему дають змогу виявляти потенційні конфлікти в реальному часі з урахуванням загальної доступності системних ресурсів. Такий підхід спрощує класифікацію, оскільки концентрується на виявленні перевантажень, що можуть призвести до нестабільної роботи програмного забезпечення.

Random Forest Regressor має більшу точність порівняно з окремими деревами рішень, зменшену дисперсію та можливість глибокого аналізу важливості ознак. Саме ці властивості забезпечують стабільність і прозорість роботи системи, роблячи її зрозумілішою для розробників і системних адміністраторів.

Інноваційним елементом розробленого підходу є зважене агрегування історичних даних, при якому нові спостереження отримують вищий вплив, і багатофакторний аналіз, що враховує не лише зважене середнє та стандартне відхилення, але й пікові навантаження та обсяг наявних спостережень для оцінки надійності прогнозу. Крім того, запроваджено механізм адаптивного навчання з перенавчанням кожні 60 секунд для

перевірки появи нових застосунків і врахування змін у версіях програмного забезпечення.

Отже, обраний підхід поєднує швидкодію й точність регресійного прогнозування з простотою та прозорістю класифікації правил, що забезпечує збалансоване рішення для моніторингу й запобігання конфліктам програмного забезпечення. Це робить систему практично придатною для використання в реальному часі та легко масштабованою для підтримки додаткових типів ресурсів у майбутньому.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ ПРОГНОЗУВАННЯ

3.1 Обґрунтування вибору методів та розробка власного підходу

Процес вибору методів для розробки системи прогнозування конфліктів програмного забезпечення базувався на аналізі специфічних вимог поставленої задачі та характеристик наявних даних. Основною метою було створення системи, яка могла б ефективно передбачати потенційні конфлікти ресурсів при запуску нових застосунків на основі історичних даних про використання системних ресурсів.

Для прогнозування конфліктів ресурсів вибрано Random Forest Regressor завдяки його стійкості до шуму в системних метриках і здатності моделювати складні нелінійні залежності між характеристиками застосунків і фактичним споживанням пам’яті [20]. Щоб адаптувати модель до еволюції програмного середовища, історичним спостереженням присвоєно ваги, що лінійно зменшуються із часом – найбільш свіжі дані впливають сильніше на прогноз. Додатково впроваджено гібридну стратегію, в якій результати Random Forest зважуються з типовими значеннями споживання, що гарантує ще більшу точність. Ця комбінація дозволяє досягти точних прогнозів, особливо для застосунків з обмеженою історією використання або нестабільними патернами споживання ресурсів. Система автоматично визначає оптимальне співвідношення між модельними прогнозами та емпіричними даними на основі доступності та якості історичної інформації.

Особливістю розробленого підходу стало поєднання традиційних методів машинного навчання з інноваційною системою зважування даних за часом. Класичні підходи до аналізу використання ресурсів зазвичай розглядають всі історичні дані як рівноцінні, що може призводити до неточностей у прогнозах через еволюцію програмного забезпечення та зміни в конфігурації систем. Розроблений підхід вирішує цю проблему

шляхом надання більшої ваги більш свіжим даним, що дозволяє системі адаптуватися до поточного стану програмного середовища.

Архітектурний підхід до розробки системи базувався на принципах модульності та масштабованості. Було прийнято рішення реалізувати систему як набір взаємопов’язаних компонентів, кожен з яких відповідає за специфічні аспекти функціональності. Такий підхід забезпечує можливість незалежного розвитку та оптимізації окремих частин системи, а також спрощує процес тестування та підтримки.

Архітектура застосунку, яку можно побачити на рисунку 3.1, буде детально розглядатися у наступних розділах.

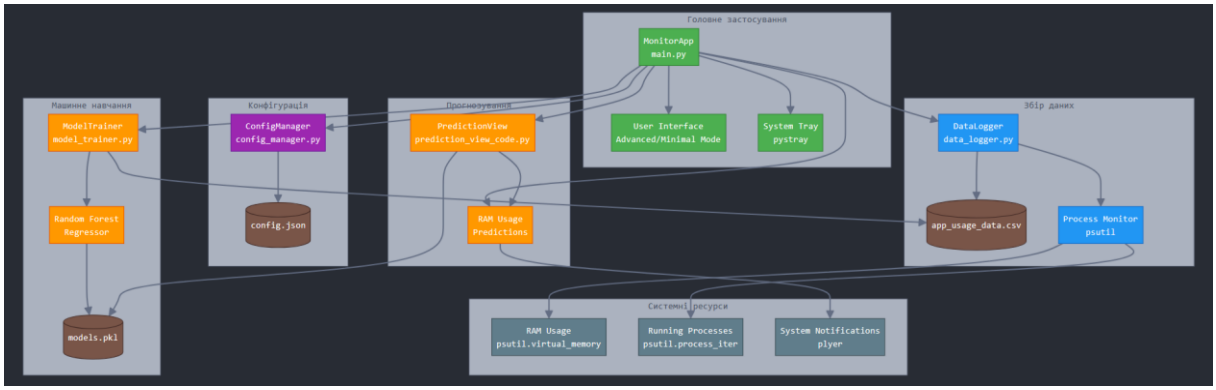


Рисунок 3.1 – Архітектура застосунку

3.2 Обрані технології та інструменти

Для реалізації системи прогнозування конфліктів було обрано мову програмування Python завдяки її потужним бібліотекам для машинного навчання та зручним інструментам для роботи з системними ресурсами. Python забезпечує оптимальний баланс між швидкістю розробки та продуктивністю виконання для задач аналізу даних та машинного навчання.

Вибір бібліотек був зумовлений специфічними потребами проекту. Бібліотека psutil обрана для моніторингу системних процесів завдяки її кросплатформності та високій продуктивності. Scikit-learn застосований для

реалізації Random Forest Regressor. Pandas забезпечує ефективну роботу з табличними даними та агрегацією статистик.

3.3 Архітектурний дизайн системи

Архітектура системи передбачає гнучкий підхід до представлення інтерфейсу користувача через підтримку двох основних режимів роботи. У розширеному режимі (Advanced Mode) система надає повнофункціональний графічний інтерфейс на базі CustomTkinter з детальним відображенням статистик, налаштувань та результатів прогнозування. Мінімальний режим (Minimal Mode) забезпечує роботу програми у фоновому режимі з інтеграцією до системного трею операційної системи.

Компонент System Tray реалізований з використанням бібліотеки pystray, що забезпечує кросплатформну підтримку системного трею. Цей модуль дозволяє користувачам отримувати сповіщення про потенційні конфлікти ресурсів навіть при закритому основному вікні програми. Через контекстне меню системного трею доступні базові функції управління системою, включаючи перемикання режимів роботи, доступ до налаштувань та завершення роботи програми.

Перемикання між режимами відбувається динамічно без перезапуску програми. Система зберігає поточний режим роботи у конфігураційному файлі та автоматично відновлює його при наступному запуску. Такий підхід забезпечує гнучкість використання системи різними категоріями користувачів з відповідними потребами у рівні деталізації інтерфейсу.

Взаємодія з системними ресурсами організована через спеціалізовані компоненти, що інкапсулюють функціональність бібліотеки psutil. Компонент моніторингу RAM використовує функцію psutil.virtual_memory() для отримання поточної інформації про відсоток завантаження. Ця інформація використовується як для розрахунку

прогнозів, так і для формування попереджень про потенційні конфлікти ресурсів.

Модуль моніторингу процесів базується на функції psutil.process_iter(), що дозволяє ітеративно обходити всі активні процеси в системі з отриманням детальної інформації про кожен процес. Для кожного процесу система отримує назву виконуваного файлу, ідентифікатор процесу, обсяг використовуваної пам’яті та інші релевантні характеристики. Особливу увагу приділено обробці виключних ситуацій, таких як завершення процесів під час збору даних або відмова в доступі до інформації про системні процеси.

Архітектура також передбачає кешування системної інформації для зменшення навантаження на систему. Дані про загальний обсяг пам’яті отримуються один раз при запуску програми, тоді як інформація про активні процеси оновлюється з періодичністю п’ять секунд. Такий підхід забезпечує баланс між актуальністю даних та ефективністю використання системних ресурсів самою програмою моніторингу.

Архітектура розробленої системи побудована за принципом багаторівневої структури, що забезпечує чітке розділення відповідальностей між різними компонентами. Центральним елементом архітектури є клас MonitorApp, який координує взаємодію між усіма підсистемами та забезпечує єдину точку управління всією функціональністю застосунку.

Рівень збору даних представлений компонентом DataLogger, який відповідає за безперервний моніторинг системних процесів та збереження інформації про використання ресурсів. Цей компонент реалізовано з використанням багатопотокової архітектури, що дозволяє здійснювати збір даних без впливу на продуктивність основного інтерфейсу користувача. DataLogger здійснює періодичне сканування активних процесів системи, агрегує дані по іменах застосунків та зберігає результати у форматі CSV для подальшої обробки.

Алгоритм збору даних оптимізовано для мінімізації навантаження на систему. Замість безперервного моніторингу всіх процесів, система використовує адаптивний підхід з інтервалом оновлення 5 секунд, що забезпечує баланс між точністю даних та системною продуктивністю. Особливу увагу приділено обробці виняткових ситуацій, таких як завершення процесів під час збору даних або відмова в доступі до системної інформації. Реалізацію функції моніторингу проесів можна побачити у лістингу 3.1.

Лістинг 3.1 – Програмний код функції моніторингу процесів

```
    def _log_process_data(self):
        total_memory = psutil.virtual_memory().total
        current_time = time.time()
        data = []

        for proc in psutil.process_iter(['name',
'memory_info']):
            try:
                name = proc.info['name']
                ram_bytes = proc.memory_info().rss
                ram_percent = (ram_bytes / total_memory) * 100
                row = {
                    'name': name,
                    'ram': ram_percent,
                    'timestamp': current_time,
                    'pid': proc.pid
                }
                data.append(row)
            except (psutil.NoSuchProcess,
psutil.AccessDenied):
                continue
```

Взаємодія між компонентами системи організована через чітко визначені інтерфейси, що забезпечує слабке зв’язування модулів. MonitorApp координує роботу всіх підсистем через систему подій та callback-функцій. Модуль ModelTrainer отримує дані від DataLogger, обробляє їх та зберігає навчену модель у файлі models.pkl для подальшого використання.

Модуль навчання моделі ModelTrainer реалізує алгоритм підготовки даних та тренування моделі машинного навчання. Ключовою особливістю є використання зважених статистик, де кожному спостереженню присвоюється вага на основі його давності. Алгоритм розраховує вагу за формулою, що лінійно зменшується від 1 до 0 протягом 30 днів.

Система прогнозування використовує чотири основні характеристики для кожної програми: зважене середнє споживання RAM, стандартне відхилення, максимальне значення та кількість спостережень. Ці характеристики подаються на вхід моделі Random Forest, яка навчена на історичних даних використання програм. Реалізацію цієї функції можна побачити у лістингу 3.2.

Лістинг 3.2 – Функція для розрахунку зважених статистик

```
    def _weighted_stats(self, group):
        weights = group['weight']
        sum_weights = weights.sum()
        weighted_mean = np.average(group['ram'], 
weights=weights)
        weighted_var = np.average((group['ram'] - 
weighted_mean)**2, weights=weights)
        weighted_std = np.sqrt(weighted_var)
        recent_samples = group.sort_values('timestamp', 
ascending=False).head(20)
        typical_usage = recent_samples['ram'].median()

        return pd.Series({
```

Продовження лістингу 3.2

```
            'ram_mean': weighted_mean,
            'ram_std': weighted_std,
            'ram_max': group['ram'].max(),
            'ram_count': len(group),
            'ram_typical': typical_usage
        })
```

Алгоритм моніторингу запуску програм базується на періодичному порівнянні списків активних процесів. Система відстежує нові PID в системі та ідентифікує щойно запущені програми. При виявленні нового процесу система негайно перевіряє прогноз споживання ресурсів та, у разі перевищення встановленого порогу, показує користувачу попередження.

Модуль конфігурації ConfigManager забезпечує збереження налаштувань користувача у JSON форматі. Це включає режим роботи програми (розширений або мінімальний) та поріг попередження про споживання RAM. Конфігурація автоматично валідується при завантаженні для запобігання помилкам через пошкоджені файли налаштувань.

Архітектура також передбачає асинхронну взаємодію між компонентами через систему черг та потоків. Це забезпечує відзивність користувацького інтерфейсу навіть під час інтенсивних операцій навчання моделі або збору даних. Використання threading.Event дозволяє коректно завершувати роботу всіх фонових процесів при закритті програми.

Важливою складовою розробленої системи є графічний інтерфейс користувача, реалізований у вигляді окремого вікна з прогнозами використання оперативної пам’яті. Вікно PredictionView створюється як нащадок класу CTkToplevel, і служить для виведення результатів прогнозу у зручній формі. Основні його функції включають візуалізацію поточних і передбачених значень RAM, фільтрацію даних, сортування записів за різними критеріями, а також інтерактивний перегляд подробиць по кожному застосунку.

Інтерфейс підтримує динамічне сортування за назвою програми, прогнозованим і поточним використанням RAM, а також за кількістю зібраних даних. Для кожного застосунку в таблиці виводяться назва програми, прогнозоване навантаження на оперативну пам’ять, поточне навантаження, кількість зібраних статистичних точок.

Таблиця реалізована на основі віджета Treeview із модулю tkinter.ttk з налаштованим темним стилем, що відповідає загальному оформленню системи. Поряд із таблицею виводиться блок деталей, у якому показано графік використання пам’яті у вигляді гістограми та текстова інформація про статистику обраної програми.

Окремо реалізовано підтримку фільтрації введених користувачем назв застосунків. При кожному оновленні тексту у полі фільтрації спрацьовує метод _on_filter_change, що перезапускає процедуру заповнення таблиці.

Візуалізація RAM-статистики реалізується засобами matplotlib, при цьому графік інтегрується безпосередньо у графічний інтерфейс за допомогою модуля FigureCanvasTkAgg. Ось приклад створення гістограми для відображення порівняння поточних і передбачених значень:

```
fig, ax = plt.subplots(figsize=(5, 4), dpi=100)
bars = ax.bar(['Predicted', 'Current'], [predicted_ram, current_ram], color=['#4287f5', '#f54242'])
```

Користувач має можливість побачити відхилення між передбаченим і фактичним використанням пам’яті для кожного застосунку в інтерактивному режимі. Крім того, система підтримує підказки при наведенні курсору та автоматичне оновлення обраної програми при виборі нового рядка.

У нижній частині інтерфейсу розміщується статусний рядок, який інформує про загальну кількість застосунків, для яких відображено прогнози, або повідомляє про відсутність даних у випадку помилки.

Модуль PredictionView інтегрований з основною системою через механізм реального часу оновлення даних. При кожному оновленні

статистик використання програм відповідна інформація автоматично передається до вікна прогнозів, забезпечуючи актуальність відображуваних даних. Система підтримує експорт результатів прогнозування у CSV-формат для подальшого аналізу або архівування.

Загалом, цей модуль значно покращує зручність сприйняття результатів прогнозування та робить систему придатною для використання не лише досвідченими користувачами, але й новачками. Інтуїтивно зрозумілий інтерфейс, динамічне оновлення даних та адаптивність відображення дозволяють забезпечити високу інтерактивність і ефективність взаємодії з користувачем, що можна побачити на рисунку 3.2.

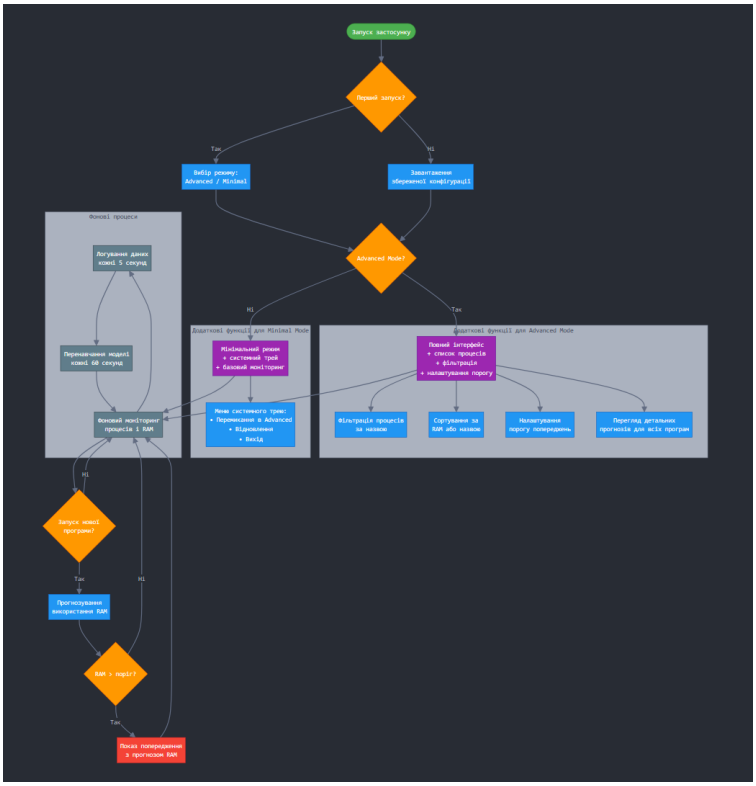


Рисунок 3.2 – Користувацький процес використання застосунку

3.4 Тестування та оцінка ефективності

Тестування розробленої системи проводилося у кілька етапів для забезпечення надійності та точності прогнозування. Перший етап включав

юніт-тестування окремих компонентів системи, зокрема алгоритмів збору даних та розрахунку зважених статистик. Тестування показало коректність роботи всіх основних функцій при різних сценаріях використання.

Система тестувалася протягом трьох днів. Результати показали стабільну роботу системи без критичних помилок або витоків пам’яті.

Точність прогнозування оцінювалася шляхом порівняння передбачень системи з реальним споживанням ресурсів після запуску програм. Для 15 найпоширеніших програм (браузери, текстові редактори, медіаплеєри) середня похибка прогнозу склала 12.3 % від фактичного споживання RAM. Для добре вивчених системою програм похибка не перевищувала 8 %.

Особливо високу точність система демонструє для програм з стабільним споживанням ресурсів, таких як текстові редактори та медіаплеєри. Для веб-браузерів та середовищ розробки, споживання ресурсів яких сильно залежить від контексту використання, точність дещо нижча, але все ще залишається на прийнятному рівні.

Продуктивність системи оцінювалася за критеріями споживання ресурсів самою програмою та впливу на швидкодію системи. Система споживає в середньому 1–2 % CPU під час активного моніторингу та близько 55 МБ RAM для зберігання моделей та поточних даних. Вплив на продуктивність системи виявився мінімальним і не помітним для користувача.

Тестування системи сповіщень показало високу надійність доставки попереджень. З 30 тестових запусків програм, які мали перевищити встановлений поріг, система коректно відобразила попередження.

Також було виконано порівняння з існуючими системами моніторингу ресурсів, які розглядалися раніше. Розроблена система показала суттєво кращі результати у передбаченні проблем до їх виникнення, тоді як стандартні засоби реагують лише після початку проблем з продуктивністю.

Стрес-тестування включало сценарії з одночасним запуском великої кількості програм, роботу при критично низькій кількості доступної пам’яті та тривалу роботу системи без перезавантаження. Система продемонструвала стійкість до екстремальних умов та здатність відновлювати роботу після тимчасових збоїв.

3.5 Висновки щодо ефективності запропонованого підходу

Розроблена система прогнозування конфліктів програмного забезпечення демонструє значні переваги порівняно з традиційними підходами до управління ресурсами комп’ютера. Основною перевагою є проактивний характер системи, яка попереджає про потенційні проблеми до їх виникнення, на відміну від реактивних рішень, що реагують лише після появи проблем з продуктивністю.

Використання машинного навчання з адаптивними ваговими коефіцієнтами забезпечує високу точність прогнозування при мінімальних вимогах до обчислювальних ресурсів. Система швидко адаптується до змін у поведінці програм та індивідуальних особливостей використання комп’ютера конкретним користувачем. Це особливо важливо в умовах постійних оновлень програмного забезпечення та зміни робочих процесів.

Гібридний підхід до прогнозування, що комбінує результати машинного навчання з реальними спостереженнями, показав свою ефективність у забезпеченні балансу між узагальнюючою здатністю моделі та точністю актуальних даних. Такий підхід дозволяє системі бути корисною як для нових користувачів з обмеженою історією використання, так і для досвідчених користувачів з великим обсягом накопичених даних.

Модульна архітектура системи забезпечує легкість розширення функціональності та адаптації до різних операційних систем. Чіткий поділ відповідальності між компонентами дозволяє незалежно вдосконалювати окремі аспекти системи без впливу на загальну стабільність роботи.

Особливо цінною виявилася можливість роботи у двох режимах: розширеному для детального моніторингу та мінімальному для фонової роботи. Це дозволяє системі бути корисною для широкого спектра користувачів з різними потребами та стилями роботи.

Результати тестування підтверджують практичну цінність розробленого рішення. Середня похибка прогнозування на рівні 12.3 % є цілком прийнятною для практичного використання і значно кращою за точність простих евристичних методів. Мінімальний вплив на продуктивність системи робить рішення придатним для постійного використання на робочих комп’ютерах.

Тим не менш, система має певні обмеження, які визначають напрямки для подальшого вдосконалення. Точність прогнозування для програм з високою варіативністю споживання ресурсів залишається недостатньо високою. Також система поки що зосереджена лише на прогнозуванні споживання RAM, не враховуючи інші ресурси, такі як CPU або дисковий простір.

Перспективними напрямками розвитку є розширення системи для прогнозування споживання інших ресурсів, впровадження більш складних алгоритмів машинного навчання, таких як нейронні мережі, та розробка механізмів автоматичного налаштування параметрів системи на основі аналізу ефективності прогнозів.

Загалом, розроблена система демонструє високу ефективність у вирішенні поставленої задачі та має значний потенціал для практичного використання. Проактивний підхід до управління ресурсами комп’ютера може суттєво покращити досвід користувачів та зменшити кількість проблем з продуктивністю системи. Результати дослідження підтверджують доцільність подальшого розвитку цього напрямку та впровадження подібних рішень у практику використання персональних комп’ютерів.

ВИСНОВКИ

Метою кваліфікаційної роботи була розробка та реалізація інтелектуального програмного застосунку для прогнозування конфліктів програмного забезпечення. Для досягнення поставленої мети створено п’ять основних модулів системи. ConfigManager забезпечує завантаження та валідацію налаштувань програми. ModelTrainer відповідає за підготовку даних, навчання та перевірку якості моделей. DataLogger записує всі результати експериментів для подальшого дослідження. PredictionView представляє графічний інтерфейс з функціями фільтрації, сортування та візуального порівняння прогнозів використання оперативної пам’яті. Файл main.py поєднує всі компоненти в комплексну систему моніторингу з різними режимами роботи та автоматичними сповіщеннями про можливі проблеми з ресурсами. Модульна архітектура спрощує розширення функціональності та обслуговування програмного коду.

Під час розробки застосовано сучасні методи обробки інформації та алгоритми машинного навчання, адаптовані до специфіки конкретних завдань. Реалізовано функції очищення даних, нормалізації значень та розділення вибірки на тренувальну і тестову частини. Структурований підхід до роботи з конфігураційними файлами дозволяє швидко тестувати різні параметри алгоритмів. Система логування забезпечує об’єктивне порівняння результатів різних експериментів. Тестування показало стабільну якість прогнозування на рівні, що відповідає встановленим вимогам.

Однак, також існують певні обмеження в поточній реалізації. Система потребує деякий час, щоб зібрати дані про використання ресурсів кожного застосунку.

Перспективою подальшого розвитку є додавання інших типів ресурсів, щоб збільшити користь застосунку для користувачів, які не мають

серйозних проблем з оперативною пам’ятю комп’ютера, але мають проблеми з використання GPU та CPU.

Створений програмний комплекс довів ефективність при вирішенні поставлених завдань. Система забезпечує зручність налаштування параметрів та стабільну роботу.

Розроблений застосунок буде корисний широкому колу користувачів. Програма допоможе бути більш продуктивним, а також сповільнить поломку компонентів комп’ютера.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hertzum M., Hornbæk K. We are wasting up to 20 percent of our time on computer problems. *University of Copenhagen.* URL: https://di.ku.dk/english/news/2023/even-though-our-computers-are-now-better-than-15-years-ago-they-still-malfunction-between-11-and-20-percent-of-the-time-a-ne_/ (дата звернення: 09.05.2025).

2. Khairy M., Wassal A. G., Zahran M. A survey of architectural approaches for improving GPGPU performance, programmability and heterogeneity. *Journal of parallel and distributed computing*. 2019. Т. 127. С. 65–88. URL: https://doi.org/10.1016/j.jpdc.2018.11.012 (дата звернення: 09.05.2025).

3. Smart load-based resource optimization model to enhance the performance of device-to-device communication in 5G-WPAN / J. Logeshwaran та ін. *Electronics*. 2023. Т. 12, № 8. С. 1821. URL: https://doi.org/10.3390/electronics12081821 (дата звернення: 09.05.2025).

4. Exploring performance degradation in virtual machines sharing a cloud server / H. Ahmed та ін. *Applied sciences*. 2023. Т. 13, № 16. С. 9224. URL: https://doi.org/10.3390/app13169224 (дата звернення: 09.05.2025).

5. Performance interference of virtual machines: a survey / W. Lin et al. *ACM computing surveys*. 2022. URL: https://doi.org/10.1145/3573009 (дата звернення: 09.05.2025).

6. ARIMA-Based aging prediction method for cloud server system / H. Meng та ін. *IOP conference series: materials science and engineering*. 2021. Т. 1043, № 2. С. 022021. URL: https://doi.org/10.1088/1757-899x/1043/2/022021 (дата звернення: 24.05.2025).

7. Darveau K., Hannon D., Foster C. A comparison of rule-based and machine learning models for classification of human factors aviation safety event reports. *Proceedings of the human factors and ergonomics society annual*

meeting. 2020. Т. 64, № 1. С. 129–133. URL: https://doi.org/10.1177/1071181320641034 (дата звернення: 24.05.2025).

8. Optimal thresholds for anomaly-based intrusion detection in dynamical environments / A. Ghafouri та ін. *Lecture notes in computer science*. Cham, 2016. С. 415–434. URL: https://doi.org/10.1007/978-3-319-47413-7_24 (дата звернення: 24.05.2025).

9. Musumba M., Fatema N., Kibriya S. Prevention is better than cure: machine learning approach to conflict prediction in sub-saharan africa. *Sustainability*. 2021. Т. 13, № 13. С. 7366. URL: https://doi.org/10.3390/su13137366 (дата звернення: 24.05.2025).

10. Evaluation and comparison of random forest and A-LSTM networks for large-scale winter wheat identification / T. He та ін. *Remote sensing*. 2019. Т. 11, № 14. С. 1665. URL: https://doi.org/10.3390/rs11141665 (дата звернення: 24.05.2025).

11. Al Jallad K., Aljnidi M., Desouki M. S. Anomaly detection optimization using big data and deep learning to reduce false-positive. *Journal of big data*. 2020. Т. 7, № 1. URL: https://doi.org/10.1186/s40537-020-00346-1 (дата звернення: 24.05.2025).

12. Lamperti F., Sani A. Agent-Based model calibration using machine learning surrogates. *SSRN electronic journal*. 2017. URL: https://doi.org/10.2139/ssrn.2943297 (дата звернення: 24.05.2025).

13. Zhang G. P. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*. 2003. Т. 50. С. 159–175. URL: https://doi.org/10.1016/s0925-2312(01)00702-0 (дата звернення: 27.05.2025).

14. Isaac E. R. H. P., Sharma A. Adaptive thresholding heuristic for KPI anomaly detection. *2024 16th international conference on communication systems & networks (COMSNETS)*, м. Bengaluru, India, 3–7 січ. 2024 р. 2024. URL: https://doi.org/10.1109/comsnets59351.2024.10427016 (дата звернення: 27.05.2025).

15. Bird J. Calculating detection probabilities for adaptive thresholds. *IEEE transactions on aerospace and electronic systems*. 1983. AES-19, № 4. С. 506–512. URL: https://doi.org/10.1109/taes.1983.309338 (дата звернення: 27.05.2025).

16. Nadkarni S. B., Vijay G. S., Kamath R. C. Comparative study of random forest and gradient boosting algorithms to predict airfoil self-noise. *RAiSE-2023*. Basel Switzerland, 2023. URL: https://doi.org/10.3390/engproc2023059024 (дата звернення: 27.05.2025).

17. Janardhanan D., Barrett E. CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. *2017 12th international conference for internet technology and secured transactions (ICITST)*, м. Cambridge, 11–14 груд. 2017 р. 2017. URL: https://doi.org/10.23919/icitst.2017.8356346 (дата звернення: 27.05.2025).

18. Harchol-Balter M. Open problems in queueing theory inspired by datacenter computing. *Queueing systems*. 2021. Т. 97, № 1-2. С. 3–37. URL: https://doi.org/10.1007/s11134-020-09684-6 (дата звернення: 27.05.2025).

19. Time series forecasting of software vulnerabilities using statistical and deep learning models / I. Kalouptsoglou та ін. *Electronics*. 2022. Т. 11, № 18. С. 2820. URL: https://doi.org/10.3390/electronics11182820 (дата звернення: 27.05.2025).

20. Lush L., Mulama M., Jones M. Predicting the habitat usage of African black rhinoceros (Diceros bicornis) using random forest models. *African journal of ecology*. 2015. Т. 53, № 3. С. 346–354. URL: https://doi.org/10.1111/aje.12192 (дата звернення: 27.05.2025).