Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

за спеціальності 121 – Інженерія програмного забезпечення

На тему: Дослідження впровадження алгоритмів машинного навчання для автоматизації процесів управління проектами

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-24м
_____/М.С.Чепурко/

Керівник
кваліфікаційної роботи _____ /Н.Н.Шаповалова/

Завідувач кафедри _____ /А.М.Стрюк/

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М.Стрюк

«__» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-24м Чепурко Максиму Сергійовичу

1. Тема: Дослідження впровадження алгоритмів машинного навчання для автоматизації процесів управління проектами. Затверджено наказом по КНУ №__ від «_» _____ 2025р.
2. Термін подання студентом закінченої роботи : «_» _____ 2025р.
3. Вихідні дані по роботі: У дослідженні використано синтетично підготовлені дані, що відображають типові параметри управління проєктами - завдання, пріоритети, ризики, завантаженість персоналу та часові показники. Алгоритми реалізовано на Python із використанням бібліотек TensorFlow і scikit-learn у середовищах PyCharm та Google Colab.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): Пояснювальна записка містить опис мети та актуальності дослідження, аналіз застосування методів машинного навчання, моделювання процесів управління проєктами, розроблення архітектури програмного рішення та узагальнення результатів експериментів.

5. Перелік ілюстративного матеріалу: Ілюстративний матеріал включає структурну схему системи, блок-схему алгоритму, діаграми моделювання, приклади роботи програмного прототипу та графіки результатів прогнозування.

УМОВНІ ПОЗНАЧЕННЯ ТА СКОРОЧЕННЯ

1. **ML (Machine Learning)** - це клас методів штучного інтелекту (ШІ), що поєднує підходи та технології для створення алгоритмів, здатних самостійно навчатися на основі даних.
2. **AI (Artificial Intelligence)** - сукупність технологій та методів, спрямованих на створення систем, здатних виконувати завдання, які зазвичай потребують людського інтелекту: аналіз, прогнозування, класифікація.
3. **API (Application Programming Interface)** - інтерфейс взаємодії між програмними компонентами, що забезпечує стандартизований обмін даними між системами, зокрема між веб-інтерфейсом та ML-моделлю.
4. **UI (User Interface)** - інтерфейс користувача, сукупність елементів та взаємодій, через які користувач працює з програмною системою чи веб-додатком.
5. **Dataset** - впорядкована сукупність даних, що використовується для навчання, валідації та тестування моделей машинного навчання.

РЕФЕРАТ

АЛГОРИТМИ МАШИННОГО НАВЧАННЯ, АВТОМАТИЗАЦІЯ УПРАВЛІННЯ ПРОЄКТАМИ, ОЦІНКА РИЗИКІВ, ПРИОРИТИЗАЦІЯ ЗАДАЧ, ПРОГНОЗУВАННЯ ПРОБЛЕМ, РЕСУРСИ ПЕРСОНАЛУ, PYTHON.

Пояснювальна записка: 109 с., 7 табл., 31 рис., 1 дод., 21 джерело.

У сучасних умовах стрімкого розвитку IT сфери, керування проєктами з кожним роком стає все складнішим і вимагає значно більше ресурсів та залучення великих команд спеціалістів. Сучасні проєкти часто мають багаторівневу структуру, різноманітні завдання та високий рівень невизначеності, що значно ускладнює їх ефективне управління. Кожен етап проєкту - від планування до реалізації - вимагає постійного контролю і коригування на основі великої кількості змінних, таких як обсяг робіт, ресурси, терміни та ризики.

Враховуючи ці складнощі, необхідність впровадження новітніх технологій для автоматизації управлінських процесів стає надзвичайно важливою. Алгоритми машинного навчання, здатні обробляти та аналізувати великі обсяги даних, можуть значно спростити ці процеси, автоматизуючи етапи, які традиційно вимагають великих зусиль з боку менеджерів проєктів.

В ході дослідження будуть проаналізовані існуючі методи машинного навчання, які застосовуються для автоматизації управлінських процесів.

В кінцевому результаті, це дозволить суттєво покращити загальну ефективність управлінських практик та скоротити витрати часу і ресурсів на вирішення рутинних задач.

ABSTRACT

MACHINE LEARNING ALGORITHMS, PROJECT MANAGEMENT AUTOMATION, RISK ASSESSMENT, TASK PRIORITIZATION, PROBLEM FORECASTING, HUMAN RESOURCES, PYTHON.

Thesis in 109 p., 7 table, 31 figure, 1 appendix, 21 source.

In today's rapidly evolving IT industry, project management is becoming more complex every year and requires significantly more resources and the involvement of large teams of specialists. Modern projects often have a multi-level structure, diverse tasks, and a high level of uncertainty, which makes them difficult to manage effectively. Each stage of the project - from planning to implementation - requires constant monitoring and adjustment based on a large number of variables, such as scope, resources, timing, and risks.

Given these complexities, the need to introduce the latest technologies to automate management processes becomes extremely important. Machine learning algorithms capable of processing and analyzing large amounts of data can greatly simplify these processes by automating steps that traditionally require a lot of effort on the part of project managers.

The study will analyze existing machine learning methods used to automate management processes.

Ultimately, this will significantly improve the overall efficiency of management practices and reduce the time and resources spent on solving routine tasks.

ЗМІСТ

ВСТУП

Постійне зростання і трансформація IT-індустрії ставлять нові виклики перед сферою управління проектами. Оскільки проекти стають дедалі складнішими, управління ними вимагає більш витончених підходів і кращої координації між різними командами фахівців. Сучасні IT-проекти часто є багаторівневими, включають численні взаємопов'язані завдання, високий рівень невизначеності та швидкозмінні вимоги. Ці фактори роблять ефективне управління проектами не лише логістичним викликом, але й стратегічною необхідністю.

Традиційних інструментів і методів управління вже недостатньо, щоб впоратися зі зростаючою складністю і динамічністю сучасних проектів. Кожен етап життєвого циклу проекту - ініціація, планування, виконання, моніторинг і закриття - вимагає контролю в режимі реального часу, постійної адаптації та прийняття рішень на основі великих і мінливих наборів даних. Керівники проектів повинні одночасно враховувати безліч параметрів, включаючи часові рамки, розподіл ресурсів, вартісні обмеження і зниження ризиків, і все це при збереженні високих рівнів продуктивності і ефективності.

У цьому контексті інтеграція передових технологій, зокрема, машинного навчання (ML), є перспективним рішенням. Алгоритми машинного навчання можуть аналізувати великі обсяги даних, пов'язаних з проектом, виявляти закономірності, прогнозувати результати та оптимізувати процеси прийняття рішень. Автоматизуючи рутинні та повторювані завдання, такі як планування ресурсів, оцінка ризиків та відстеження продуктивності, інструменти ML зменшують навантаження на менеджерів проектів і дозволяють їм зосередитися на стратегічному плануванні та інноваціях.

У цій кваліфікаційній роботі досліджується, як машинне навчання може бути ефективно застосоване для автоматизації різних аспектів управління проектами, зокрема приділено особливу увагу аспекту ризик менеджменту. Вона містить огляд

ключових методів ML, що використовуються в цій галузі, оцінку їх практичного застосування, а також розглядає конкретні приклади та інструменти, які ілюструють їх переваги. Мета дослідження - оцінити, як використання технологій машинного навчання може підвищити ефективність, точність і гнучкість процесів управління проектами, що призведе до кращих результатів проекту при скороченні витрат часу і ресурсів.

Розглядаючи ці питання, це дослідження сприяє глибшому розумінню того, як технологічний прогрес може змінити традиційні практики управління та підтримати сталий розвиток складних IT-проектів у середовищі, де все більше керують дані.

1 СУЧАСНИЙ СТАН І АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

1.1 Актуальність теми дослідження

У сучасній цифровій економіці управління проектами більше не є суто адміністративною функцією - воно перетворилося на динамічну та керовану даними дисципліну, яка безпосередньо впливає на успіх та конкурентоспроможність організацій.

Зростаюча складність IT-проектів у поєднанні зі збільшенням обсягу доступних даних створила гостру потребу в нових методах та інструментах, які можуть покращити процес прийняття рішень та зменшити навантаження на людину. У цьому контексті використання машинного навчання (ML) в управлінні проектами є не просто актуальним - воно є стратегічно необхідним.

Ключові фактори, які підкреслюють актуальність цієї теми:

1. Швидке зростання обсягів даних у проектному середовищі

   Згідно з дослідженням IDC [1], очікується, що до 2026 року обсяг цифрових даних у світі зросте до 175 зеттабайт. Значна частка цих даних надходить з корпоративних систем, включаючи відстеження проектів, розподіл ресурсів та інструменти для спільної роботи команд. Керувати такими обсягами вручну вже неможливо;

2. Обмеження традиційних підходів до управління проектами

   Класичні моделі, такі як Waterfall чи навіть Agile, хоча й ефективні в певних сценаріях, значною мірою покладаються на досвід та інтуїцію проектних менеджерів. Вони часто не здатні динамічно адаптуватися до змін у доступності ресурсів, продуктивності команди або пріоритетів зацікавлених сторін у реальному часі;

   Наприклад, затримки у виконанні завдань через непередбачені вузькі місця можуть залишатися непоміченими, поки вони не вплинуть на терміни. Моделі

прогнозування на основі ML можуть виявити такі ризики на ранніх стадіях, аналізуючи історичні та поточні дані.

3. Автоматизація як інструмент оптимізації ресурсів

   За даними McKinsey & Company, до 45% завдань, які виконують менеджери проектів, можна автоматизувати за допомогою існуючих технологій. До них відносяться оптимізація графіку, прогнозування бюджету та оцінка ризиків. Автоматизувавши рутинні операції, команди можуть перенаправити свою увагу на творчі та стратегічні завдання, тим самим підвищуючи показники успішності проектів;

4. Зростання попиту на інтелектуальні інструменти управління проектами

   За прогнозами, до 2028 року світовий ринок інструментів управління проєктами на основі штучного інтелекту досягне 4,3 мільярда доларів США [3]. Це свідчить про широке визнання потреби в передових рішеннях, здатних впоратися зі складнощами сучасних проектів, особливо у сфері великомасштабних IT-розробок і розробки програмного забезпечення.

1.2 Цілі та завдання дослідження

Основною метою кваліфікаційної роботи є:

Проаналізувати, оцінити та продемонструвати потенціал методів машинного навчання для автоматизації процесів управління проектами з акцентом на підвищення точності прийняття рішень, зменшення ризиків та підвищення загальної ефективності проектів в IT-середовищі.

Для досягнення цієї мети були визначені наступні конкретні завдання:

1. Провести теоретичний аналіз методологій управління проектами та визначити поточні проблеми в управлінні складними IT-проектами [1];

2. Вивчити існуючі алгоритми машинного навчання, застосовні до завдань управління проектами, таких як предиктивна аналітика, оптимізація ресурсів і виявлення аномалій [5];

3. Класифікувати та оцінити найпоширеніші моделі ML з точки зору їхньої придатності для автоматизації конкретних управлінських завдань (наприклад, регресійні моделі для оцінки часових рамок, кластеризація для групування завдань) [6];

4. Проаналізувати існуючі програмні інструменти та платформи, які інтегрують ML в системи управління проектами (наприклад, Microsoft Project з розширеннями AI, функції Smart Assist в Asana, автоматизація Jira з плагінами ML) [2];

5. Розробити концептуальну основу або прототип моделі, яка ілюструє, як машинне навчання може автоматизувати обраний процес в управлінні проектами (наприклад, прогнозування дедлайнів або визначення пріоритетів ризиків);

6. Оцінити переваги та обмеження інтеграції технологій машинного навчання в робочі процеси проєктів, включаючи організаційні, етичні та технічні міркування;

7. Надати практичні рекомендації для компаній та менеджерів проєктів, які прагнуть впровадити автоматизацію на основі технологій ML у свої робочі процеси, підкріплені реальними кейсами та академічними дослідженнями.

Для досягнення поставленої мети у роботі застосовано комплекс методів дослідження, що дозволяє забезпечити комплексний аналіз проблеми та розробку рекомендацій:

1. Аналіз наукових джерел та літератури – з метою виявлення сучасних підходів до автоматизації управління IT-проєктами за допомогою машинного навчання;

2. Порівняльний аналіз – для визначення переваг і недоліків різних алгоритмів ML, які використовуються в проєктному менеджменті;

3. Емпіричний метод – для апробації обраної моделі на конкретних кейсах управління IT-проєктами;

4. Моделювання – для розробки та тестування прототипу автоматизації управлінського процесу з використанням алгоритмів машинного навчання;

5. Метод системного аналізу – для оцінки впливу впроваджених рішень на ефективність управління проєктами;

6. Статистичний аналіз – для обробки даних, отриманих під час апробації моделі, та оцінки її ефективності в реальних умовах.

Очікується, що в результаті виконання кваліфікаційної роботи буде:

- Підтверджено доцільність використання алгоритмів машинного навчання для підвищення точності прийняття управлінських рішень у сфері IT-проєктів;
- Розроблено рекомендації щодо вибору конкретних ML-алгоритмів залежно від специфіки задач проєктного управління;
- Запропоновано прототип автоматизованої системи підтримки управлінських рішень з використанням алгоритмів ML;
- Продемонстровано переваги інтеграції методів машинного навчання в управління IT-проєктами на основі реальних кейсів та симуляційних експериментів.

1.3 Критичний аналіз літературних джерел за темою

Інтеграція машинного навчання (ML) в управління проектами в останні роки привертає все більшу увагу як академічних дослідників, так і практиків галузі. Однак критичний огляд існуючої літератури свідчить про те, що ця сфера є фрагментованою і все ще розвивається, де теоретичні засади та практична реалізація часто не узгоджуються між собою.

Класичні та сучасні підходи до управління проєктами добре задокументовані у фундаментальних джерелах, таких як PMBOK® Guide та методології PRINCE2. Ці фреймворки визначають структуру та принципи управління обсягом, часом, вартістю, якістю та ризиками проєкту. Однак, хоча вони надають надійні моделі для

процедурного контролю, їм бракує механізмів для динамічної обробки великомасштабних даних, що обмежує їхню адаптивність у швидкозмінному середовищі.

Потенціал ML для покращення результатів проєктів є широко визнаним. У науковій літературі подано ґрунтовні огляди алгоритмів машинного навчання — від методів керованого навчання (регресія, класифікація) до методів некерованого навчання (кластеризація, виявлення аномалій). Однак більшість таких робіт зосереджені переважно на технічних аспектах ML і рідко мають пряме застосування до практичних сценаріїв управління проєктами.У цьому напрямку було досягнуто певного прогресу. Наприклад:

1. У 2022 році запропоновано модель предиктивної аналітики для управління розкладом проєктів на основі градієнтного бустингу, що показує в середньому близько 15% приросту точності порівняно з традиційними методами оцінювання;
2. У 2021 році проведено дослідження, присвячене використанню нейронних мереж для прогнозування розподілу ресурсів та виявлення ризиків, яке підтвердило підвищення точності у проєктах з високою волатильністю;
3. У 2023 році окреслено тенденцію «AI-augmented project management», що передбачає застосування ML для автоматизації розподілу завдань, моніторингу прогресу та визначення пріоритетів.

Незважаючи на цей розвиток, багато досліджень все ще залишаються концептуальними, їм бракує експериментальної перевірки або поздовжніх тематичних досліджень. Крім того, часто ігнорується проблема інтерпретації результатів ВК у бізнес-контексті, яку часто називають «проблемою чорної скриньки».

Реальні платформи, такі як Jira, Asana, Trello і Microsoft Project, почали впроваджувати функції на основі ML, такі як автоматичні пропозиції щодо дедлайнів, аналіз настроїв на основі журналів комунікацій і прогнозоване

балансування робочого навантаження. Однак ці впровадження часто є поверхневими і призначені для загальної продуктивності, а не для прийняття стратегічних рішень на рівні проекту.

Згідно з циклом зрілості технологій для штучного інтелекту, багато функцій управління проєктами на основі штучного інтелекту все ще перебувають на стадії «піку завищених очікувань», з обмеженим розумінням їхнього довгострокового впливу на результати проєкту.

На основі цього огляду можна окреслити кілька прогалин:

1. Недостатня кількість моделей для конкретних галузей: Більшість існуючих застосувань ML в управлінні проектами є загальними і не пристосовані до складних структур IT-проектів;
2. Відсутність інтеграційних фреймворків: Не існує комплексної методології для вбудовування інструментів ML в існуючі робочі процеси управління проектами;
3. Обмеженість емпіричних досліджень: Небагато публікацій включають реальні результати розгортання, лонгітюдні дані або показники ефективності в різних галузях;
4. Етичні та людські фактори: Недостатньо вивчені такі питання, як прийняття колективом, упередженість моделей прогнозування та прозорість автоматизованих рішень.

На закінчення, хоча академічна і промислова література забезпечує міцну основу для розуміння окремих компонентів - машинного навчання, з одного боку, і управління проектами, з іншого, - існує очевидна потреба в міждисциплінарних дослідженнях, які об'єднують ці дві області. Ця робота має на меті зробити внесок у цю нову галузь шляхом критичної оцінки, систематизації та застосування методологій машинного навчання в контексті практичних завдань управління проектами.

1.4 Основні особливості предметної галузі

За останнє десятиріччя інтенсивно досліджується впровадження штучного інтелекту та машинного навчання (МН) для підтримки й автоматизації різних аспектів управління проєктами. Ідея полягає в тому, щоб доручити рутинні або аналітичні завдання комп'ютерним алгоритмам, здатним аналізувати великі обсяги даних та виявляти приховані закономірності для обґрунтування рішень. Сучасні дослідження показують, що інструменти ШІ можуть підвищувати ефективність проєктного менеджменту, зменшувати кількість помилок і упередженостей та давати точніші прогнози розвитку проєкту. Розглянемо наукові роботи, присвячені застосуванню різних алгоритмів МН на основних етапах життєвого циклу проєкту – від планування до контролю та завершення.

Планування ІТ-проєкту – це визначення оптимальної структури робіт, ресурсів і графіку їх виконання. Для підтримки цих задач дослідники застосовують методи штучного інтелекту, зокрема алгоритми навчання з підкріплення (reinforcement learning). Наприклад, проблему оптимального розподілу ресурсів і календарного планування можна сформулювати як Markov Decision Process (MDP) і розв'язувати за допомогою глибокого навчання з підкріплення (Deep Q-Learning). Один з підходів продемонстрував, що алгоритм Deep Q-Network здатний навчитися політики розподілу ресурсів для проєкту з урахуванням повторних ітерацій робіт і можливості «crashing» (прискорення виконання за рахунок додаткових ресурсів). Експериментальне порівняння показало, що навчений таким чином агент ухвалює рішення не гірше за традиційні евристичні правила, оптимізуючи тривалість проєкту за обмежених ресурсів.

Інший напрям планування – вибір підходящої методології управління (традиційна каскадна чи Agile) залежно від характеристик проєкту. Тут також успішно застосовуються алгоритми класифікації. Зокрема, дослідження 2023 року показало, що модель на основі градієнтного бустингу (ансамблю рішучих дерев) може з точністю ~94% прогнозувати, який метод управління (гнучкий чи

класичний) найбільш придатний для нового проєкту. У цій роботі алгоритм Gradient Boosting перевершив SVM та k-Nearest Neighbors, особливо після застосування технік балансування вибірки (SMOTE) для врахування нерівномірності даних. Такі результати демонструють, що МН здатне підтримати менеджера на етапі стратегічного планування – від вибору методології до автоматизованого створення графіків. У перспективі, подальший розвиток моделей планування (включно з нейромережевими та підкріплювальними алгоритмами) обіцяє повніше автоматизувати складні задачі розкладу в реальному часі та перерозподілу ресурсів при змінах.

Аналіз наукових джерел останніх років свідчить, що машинне навчання проникає в усі етапи управління проєктами. Вже зараз ШІ-інструменти допомагають менеджерам проектів у стратегічному плануванні, оцінці ризиків, прогнозуванні термінів/бюджетів та підтримці роботи команд. В різних галузях – від будівництва до розробки ПЗ – успішно використовуються ML-рішення для підтримки прийняття рішень на основі даних (від систем управління знаннями до інтелектуальних дашбордів моніторингу проєктів). Автоматизація рутинних операцій за допомогою ШІ дозволяє звільнити час менеджера і команди для творчих та складних завдань, що вимагають людської експертизи [8].

Водночас, література наголошує й на викликах впровадження ШІ в проєктний менеджмент. Потрібно враховувати питання якісних даних, конфіденційності, а також небезпеку алгоритмічної упередженості у прогнозних моделях. Впровадження ML-рішень потребує зміни процесів і навчання персоналу, а ефективність моделей залежить від наявності достатнього обсягу достовірних історичних даних про проєкти. Отже, людський фактор залишається важливим: навіть найкращі алгоритми мають слугувати підтримкою для компетентного менеджера, а не повністю його замінювати. Успішні кейси показують, що синергія досвіду керівників проектів та аналітичної потужності ШІ може привести до значного підвищення ймовірності завершення проектів вчасно і з дотриманням

бюджету. Продовження досліджень в цьому напрямку – включно з пояснюваним ШІ (explainable AI) для проєктного аналізу – відкриває нові перспективи вдосконалення практики управління ІТ-проєктами у майбутньому.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ

2.1 Визначення вимог

Процес визначення вимог до системи автоматизації управління ІТ-проєктами на основі методів машинного навчання є важливим етапом проєктування. Від правильності цього етапу залежить, наскільки ефективною, продуктивною та адаптивною буде розроблена система. У відповідності до загальноприйнятих методологій системного аналізу (PMBOK Guide, 2021), вимоги поділяються на функціональні та нефункціональні. Функціональні вимоги відображають основні задачі, які система має виконувати, а нефункціональні - визначають характеристики, що впливають на якість, безпеку, продуктивність та інтеграційні можливості системи.

Таблиця 2.1 - Функціональні вимоги до системи.

Тип вимог	Опис
Функціональні	- Дослідження можливостей прогнозування термінів виконання завдань на основі аналізу історичних даних з використанням регресійних моделей. - Аналіз ризиків та потенційних відхилень за допомогою алгоритмів класифікації (наприклад, SVM, Decision Trees), що дозволить сформувати висновки щодо найбільш ризикових етапів проєкту. - Моделювання розподілу ресурсів між завданнями з використанням методів машинного навчання (наприклад, reinforcement learning), з метою оцінки можливих варіантів

	оптимізації ресурсів. - Формування рекомендацій на основі результатів кластеризації завдань, що дозволить обґрунтувати доцільність застосування певних ML-алгоритмів для аналізу залежностей між елементами проєкту. - Візуалізація отриманих даних у вигляді графіків та діаграм для наочного представлення результатів дослідження.
Нефункціональні	- Забезпечення коректності аналізу та перевірки результатів дослідження шляхом використання статистичних методів та оцінки похибок моделей. - Гнучкість методів, що досліджуються, для можливості адаптації до різних типів даних та специфіки проєктів. - Зручність подання результатів дослідження у формі інтерактивних дашбордів або звітів, що підкреслює основні тенденції та виявлені аномалії. - Верифікація результатів дослідження за допомогою експертних оцінок та порівняння з реальними даними проєктів. - Дотримання принципів академічної доброчесності при використанні даних та побудові висновків.

2.2 Аналіз використання різних алгоритмів машинного навчання

2.2.1 Регресійні алгоритми

Регресійні алгоритми - це методи машинного навчання, які використовуються для визначення зв’язку між незалежними змінними (факторами, предикторами) та залежною змінною (цільовою змінною). Основна мета регресії - прогнозування

кількісного результату на основі аналізу вхідних даних. Регресійні алгоритми дозволяють:

- Виявляти лінійні та нелінійні залежності між змінними;
- Прогнозувати значення залежної змінної для нових даних;
- Оцінювати вплив окремих факторів на кінцевий результат.

Регресійні алгоритми поділяються на кілька типів:

1. Лінійна регресія (Linear Regression);
2. Поліноміальна регресія (Polynomial Regression);
3. Регресія з регуляризацією (Ridge, Lasso);
4. Логістична регресія (Logistic Regression) - для класифікаційних задач.

У сфері управління проєктами регресійні алгоритми використовуються для прогнозування основних метрик, таких як:

- Тривалість виконання завдань - прогнозування часу, необхідного для завершення етапів проєкту, на основі обсягу робіт, кількості ресурсів, складності завдань;
- Витрати, оцінка витрат на проєкт з урахуванням попередніх даних (кількість годин, оплата праці, вартість матеріалів);
- Ризики, аналіз впливу певних факторів на ймовірність відхилення від плану (наприклад, кількість змін у вимогах, досвід команди);
- Прогнозування продуктивності команди на основі її попередніх показників (кількість виконаних завдань, час на виконання).

Лінійна регресія - це базовий алгоритм, що описує лінійну залежність між незалежною змінною X X та залежною змінною Y Y.

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{2.1}$$

де:

- Y - залежна змінна (наприклад, час виконання завдання);

- X - незалежна змінна (наприклад, кількість завдань у проєкті);
- $\beta 0$- константа (вільний член);
- $\beta 1$ - коефіцієнт нахилу (вплив X на Y);
- ε- випадкова похибка.

Якщо у нас є кілька незалежних змінних (наприклад, кількість завдань, складність завдання, досвід команди), формула багатовимірної регресії буде виглядати так:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \varepsilon$$

(2.2)

Завдання:

Керівник проєкту хоче спрогнозувати тривалість виконання завдання (у днях) на основі таких факторів:

1. Кількість завдань у проєкті.
2. Досвід команди (у роках).
3. Складність завдання (бали від 1 до 5).

Історичні дані виглядають так:

Таблиця 2.2 – Приклад історичних даних

Кількість завдань	Досвід (роки)	Складність	Час (днів)
15	2	3	5
30	5	2	7
20	3	4	6
25	4	3	7

Тоді модель виглядає так:

$$Y = 1.5 + 0.2 \times X_1 - 0.1 \times X_2 + 0.8 \times X_3$$

(2.4)

Для нового завдання з такими параметрами:

- Кількість завдань: 18
- Досвід: 3 роки
- Складність: 4

Отримуємо прогнозований час виконання:

$$Y = 1.5 + 0.2 \times 18 - 0.1 \times 3 + 0.8 \times 4$$

$$Y = 1.5 + 3.6 - 0.3 + 3.2 = 8.0 \text{ днів}$$

(2.3)

Припустимо, що ми маємо проєкт, у якому було зібрано дані про виконання завдань: кількість завдань, досвід команди та складність завдань. Використовуючи ці дані, ми побудували регресійну модель для прогнозування часу виконання завдань.

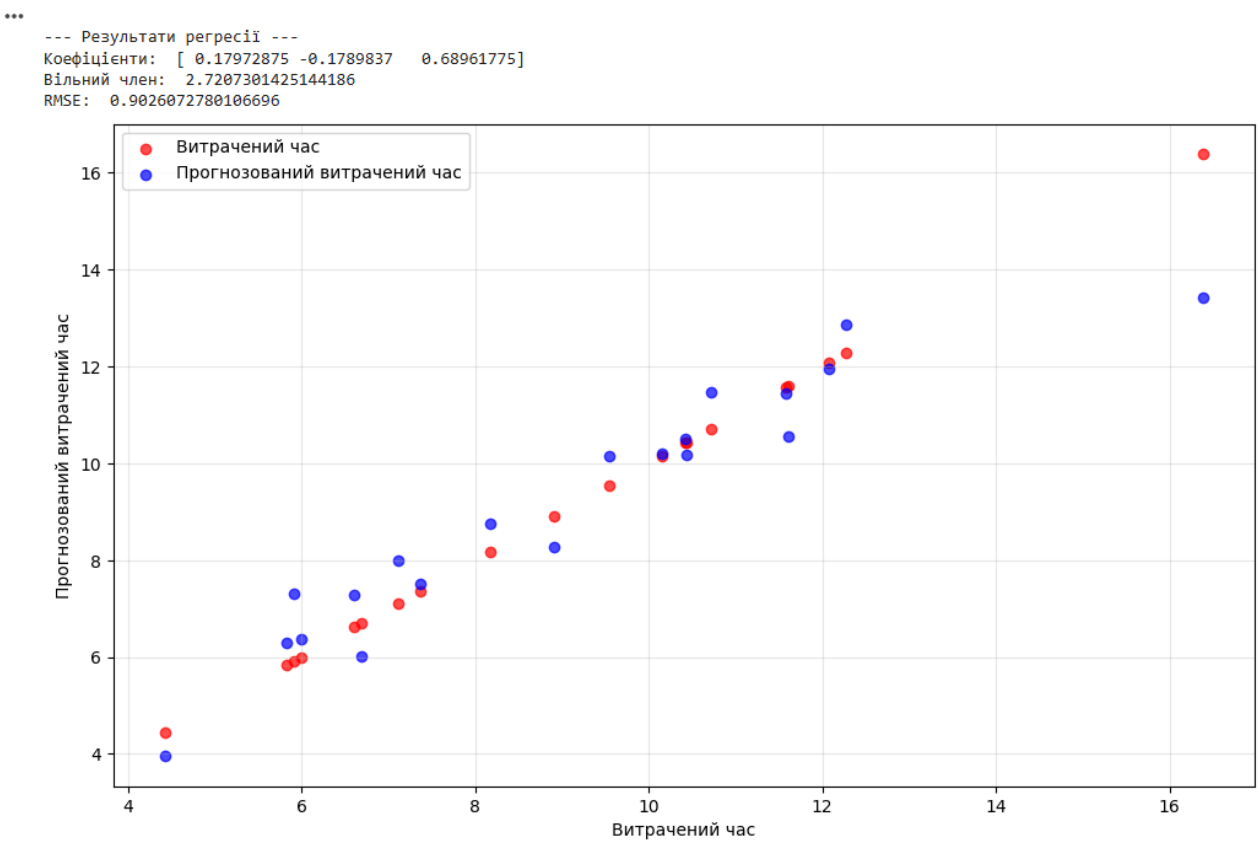


Рисунок 2.1 – Графік регресійної моделі для прогнозування часу виконання задачі

```
num_tasks = np.random.randint(1, 15, n_samples)
experience = np.random.randint(10, 15, n_samples)
complexity = np.random.randint(1, 3, n_samples)
time_spent = 1.5 + 0.2 * num_tasks - 0.1 * experience + 0.8 * complexity + np.random.normal(0, 1, n_samples)
```

Рисунок 2.2 – Зображення вхідних даних до задачі

На графіку ми бачимо дві групи точок:

- Червоні точки - це фактичний час виконання завдань (Actual Time Spent). Вони представляють реальні значення, які було зібрано на етапі аналізу даних. Вони показують, скільки днів фактично знадобилося для виконання певної кількості завдань із заданою складністю та рівнем досвіду команди.
- Сині точки - це прогнозовані значення (Predicted Time Spent), які модель регресії розрахувала на основі вхідних даних. Вони показують, який час модель вважає оптимальним для виконання завдання, враховуючи історичні дані.

Чим ближче сині точки розташовані до червоних, тим точніше модель спрогнозувала час виконання завдань. Ідеальним випадком було б повне збігання червоних та синіх точок, що означало б абсолютну точність прогнозу.

У нашому випадку, використовуючи регресійну модель, ми отримали певне відхилення між фактичними та прогнозованими значеннями. Однак, враховуючи отримане значення RMSE (Root Mean Squared Error), яке ми також розрахували в коді, можна оцінити загальну точність моделі.

При збільшенні складності задач, та зменшенні досвіду команди – прогнозувати задачі складніше:

```
--- Результати регресії ---
Коефіцієнти:  [ 0.21018594 -0.03467451  0.        ]
Вільний член:  4.178275316573339
RMSE:  1.1895061664932658
```

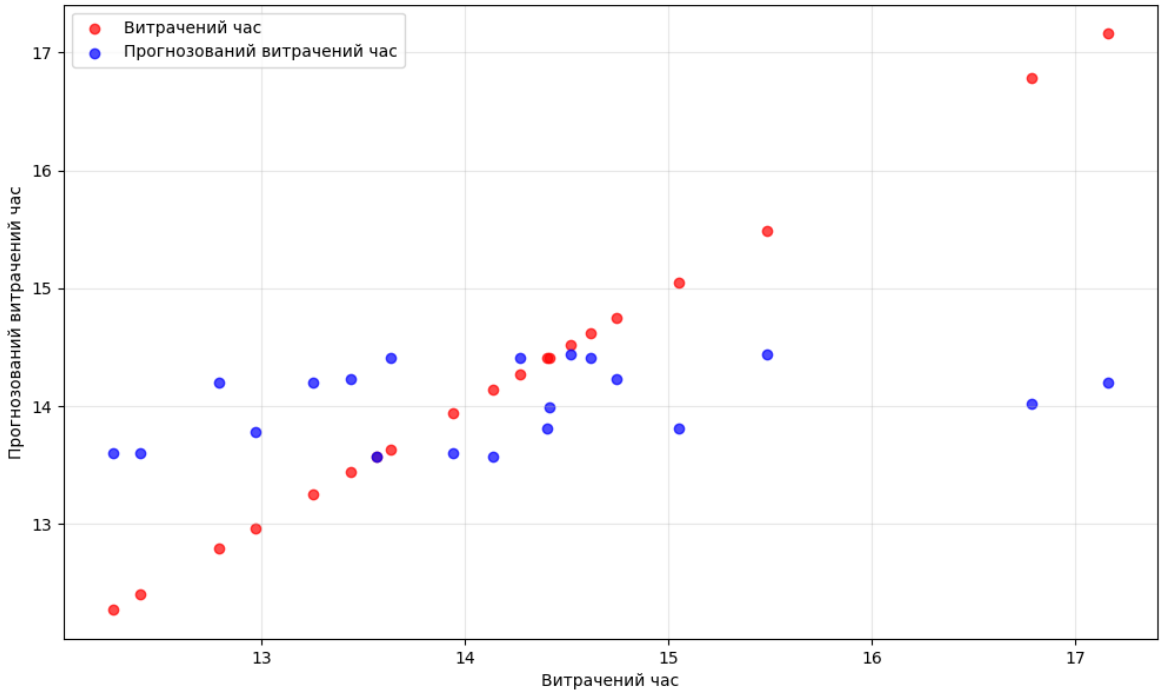


Рисунок 2.3 – Графік регресійної моделі для прогнозування часу виконання задачі

```
num_tasks = np.random.randint(25, 50, n_samples)
experience = np.random.randint(1, 3, n_samples)
complexity = np.random.randint(1, 5, n_samples)
time_spent = 1.5 + 0.2 * num_tasks - 0.1 * experience + 0.8 * complexity + np.random.normal(0, 1, n_samples)
```

Рисунок 2.4 – Зображення вхідних даних до задачі

2.2.2 Моделі часових рядів. LSTM (Long Short-Term Memory)

Припустимо, що ми аналізуємо історичні дані про виконання завдань у проєкті: скільки часу знадобилося на виконання завдань у певний період. Ми хочемо спрогнозувати майбутні часові показники на основі цих даних. Для цього використовуємо модель LSTM, яка здатна аналізувати залежності між значеннями у часових рядах.

На графіку прогнозу LSTM зазвичай можна побачити дві основні лінії:

- Червона лінія - фактичні значення (Actual Time Spent). Це дані, які ми маємо на основі попередніх спостережень. Вони показують, скільки днів реально витрачалося на виконання завдань у минулому.
- Синя лінія - прогнозовані значення (Predicted Time Spent). Це результати, отримані за допомогою LSTM. Вони показують, скільки днів модель вважає необхідним для виконання завдань у майбутньому періоді.

Чому LSTM підходить для прогнозу часових рядів?

LSTM здатна враховувати попередні стани, тобто не лише поточні значення, але й те, що відбувалося раніше. Це особливо корисно для проєктів, де спостерігаються певні циклічні або трендові залежності.

Також якщо у проєкті раніше спостерігалися затримки у виконанні завдань, модель може врахувати це і спрогнозувати подібні ситуації у майбутньому. LSTM може виділяти аномальні значення та враховувати їх при формуванні прогнозу, що дозволяє мінімізувати вплив різких змін у даних.

Наприклад, якщо модель LSTM аналізує часовий ряд із 100 днів, де на кожний день фіксується витрачений час на завдання, вона формує прогноз для наступних 10 днів. Якщо фактичний час виконання завдань різко зріс за останні 5 днів, LSTM врахує цю тенденцію та спрогнозує подібний підвищений показник на наступний період.

Таким чином, модель LSTM дозволяє прогнозувати часові показники у проєктах не лише на основі поточного стану завдання, але й з урахуванням довгострокових закономірностей у даних. Це робить її потужним інструментом для аналізу проєктних метрик, де часові ряди мають велике значення.

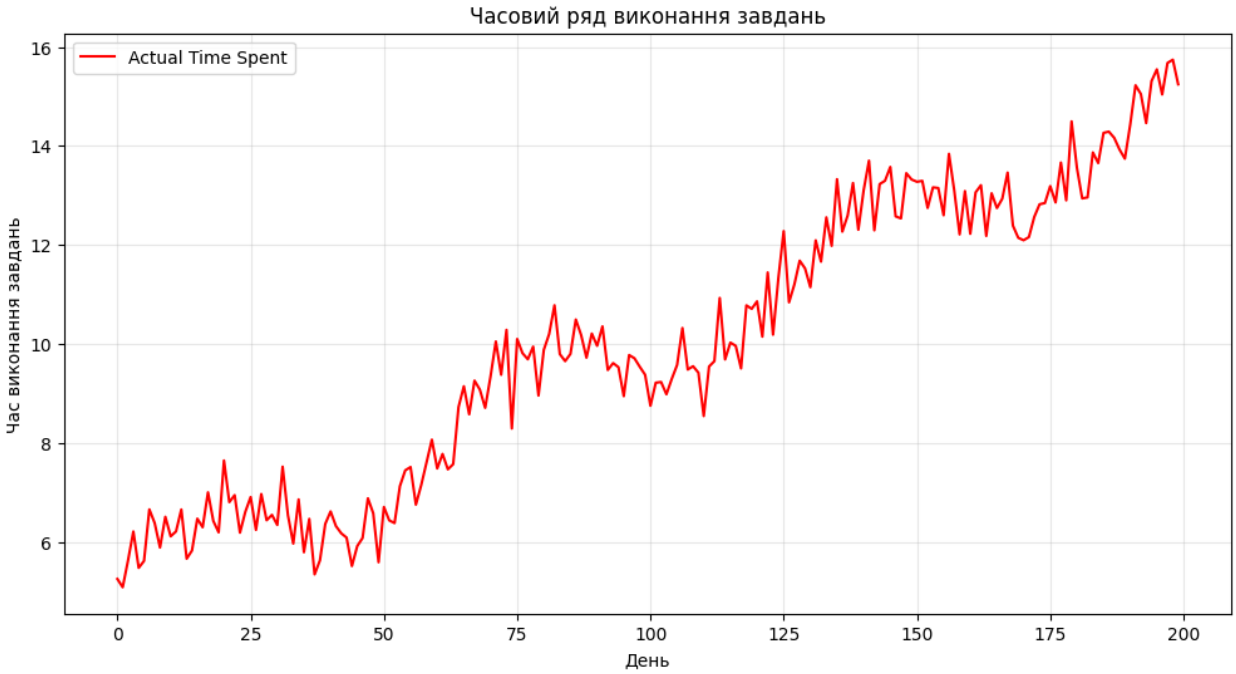


Рисунок 2.5 – Графік часового рядку виконання завдань

```
Epoch 30/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - loss: 0.4295 - val_loss: 0.2491
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 198ms/step
RMSE: 0.499089861460212
```

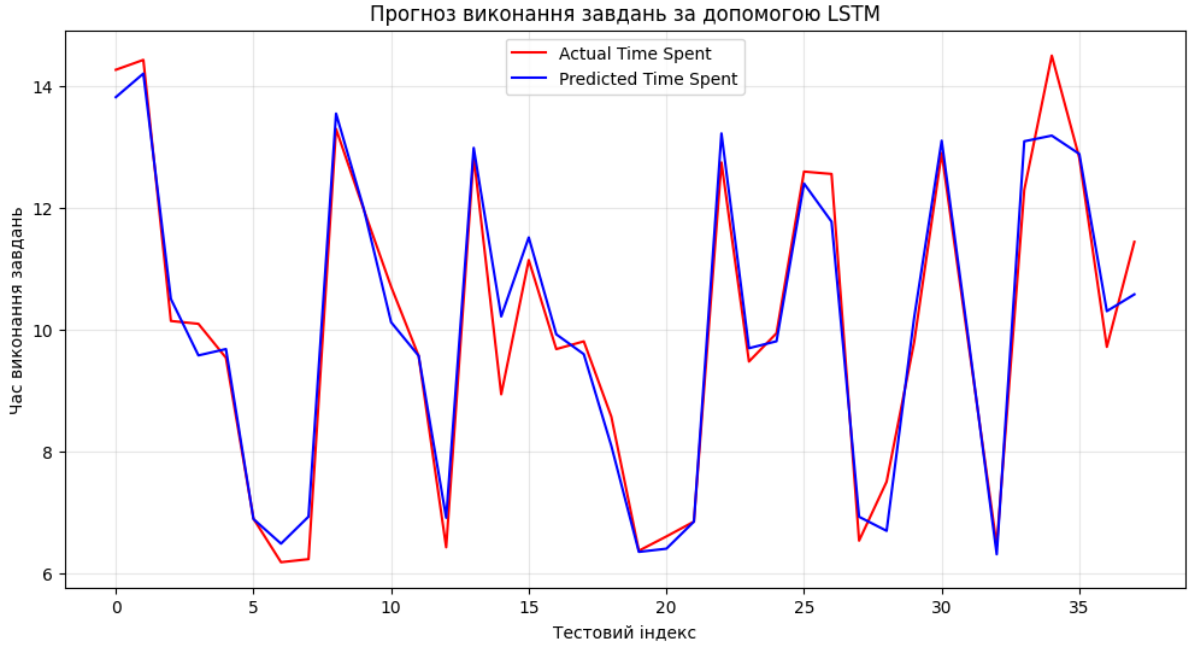


Рисунок 2.6 – Графік прогнозу виконання завдань за допомогою LSTM

У цьому прикладі використано модель LSTM (Long Short-Term Memory) для прогнозування часу виконання завдань на основі часових рядів даних. Мета - спрогнозувати майбутні значення на основі історичних даних про витрати часу на виконання завдань.

Структура даних:

- Згенеровано часовий ряд на 200 днів, що включає періодичні коливання та випадкові коливання;
- Червона лінія на графіку - фактичні значення (Actual Time Spent), які відображають реальні дані про витрати часу;
- Синя лінія - прогнозовані значення (Predicted Time Spent), які LSTM розрахувала на основі навчальних даних.

Архітектура LSTM:

- Вхідний шар: послідовність із 10 днів;
- Прихований шар: 50 нейронів;
- Вихідний шар: одне прогнозоване значення;
- Функція втрат - MSE, оптимізатор – adam;
- Навчання моделі виконувалося протягом 30 епох.

Чим ближче синя лінія до червоної, тим точніше прогноз. Відхилення можуть вказувати на нестачу навчальних даних або необхідність налаштування моделі. Модель LSTM здатна виявляти як короткострокові, так і довгострокові тенденції, що важливо для аналізу проєктних часових рядів.

Такий підхід дозволяє прогнозувати не лише конкретні часові показники, але й тенденції до перевищення строків виконання завдань.

2.2.3 Випадковий ліс (Random Forest)

Random Forest - це ансамблевий алгоритм машинного навчання, який складається з набору дерев рішень. Кожне дерево будується на випадковій

підмножині даних і приймає своє рішення. Остаточний прогноз формується як середнє (для регресії) або голосування (для класифікації) серед усіх дерев.

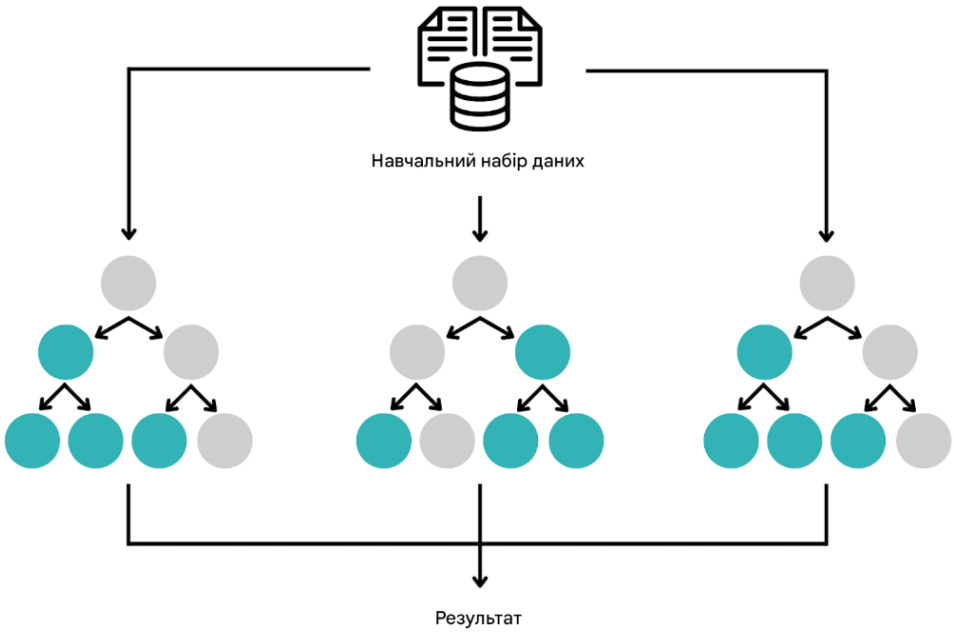


Рисунок 2.7 – Візуалізація методу Random Forest

У контексті управління ризиками, Random Forest може використовуватись для:

1. Класифікації ризиків, визначення завдань або етапів проєкту, що мають високий/середній/низький ризик зриву;
2. Прогнозування ймовірності ризику, оцінка ймовірності виникнення певного ризику на основі історичних даних про проєкти;
3. Визначення найбільш значущих факторів, що впливають на ризик (наприклад, зміна вимог, затримки поставок, кількість змін у проєкті).

У наступному прикладі було реалізовано аналіз ризиків у проєкті за допомогою алгоритму Random Forest. Для цього було згенеровано великий та реалістичний набір даних, що містить 1000 записів про різноманітні аспекти виконання завдань у проєкті. Дані включають такі змінні, як кількість змін у

вимогах, досвід команди, складність завдання, розмір команди, перевищення бюджету, час виконання, задоволеність клієнта, кількість багів, кількість критичних проблем та затримка у днях. Кожен запис містить цільову змінну - рівень ризику від 1 до 10.

Цей набір даних імітує реальну проєктну ситуацію, де можна побачити вплив різних факторів на ризик. Наприклад, велика кількість змін у вимогах або значна затримка в днях можуть суттєво збільшити рівень ризику для конкретного завдання чи етапу проєкту. Використання широкого спектра факторів дозволяє моделі навчитися виявляти як очевидні, так і приховані взаємозв'язки між змінними.

Для класифікації ризиків було обрано алгоритм Random Forest, оскільки він забезпечує високу точність і стійкість до шуму в даних. Модель складається зі 100 дерев рішень, де кожне дерево голосує за свій прогноз ризику. Остаточний прогноз визначається як середнє значення всіх прогнозів дерев. Це дозволяє зменшити ймовірність випадкових помилок та забезпечити узагальнення моделі.

Після навчання моделі було виконано прогноз для тестової вибірки, що складає 30% даних. Для оцінки якості моделі було побудовано кілька графіків.

Перший графік - Матриця помилок - демонструє розподіл правильно та неправильно класифікованих ризиків. Горизонтальна вісь показує прогнозовані ризики, а вертикальна - фактичні. Яскраво сині клітинки на діагоналі вказують на правильні прогнози, тоді як клітинки поза діагоналлю демонструють помилки класифікації.

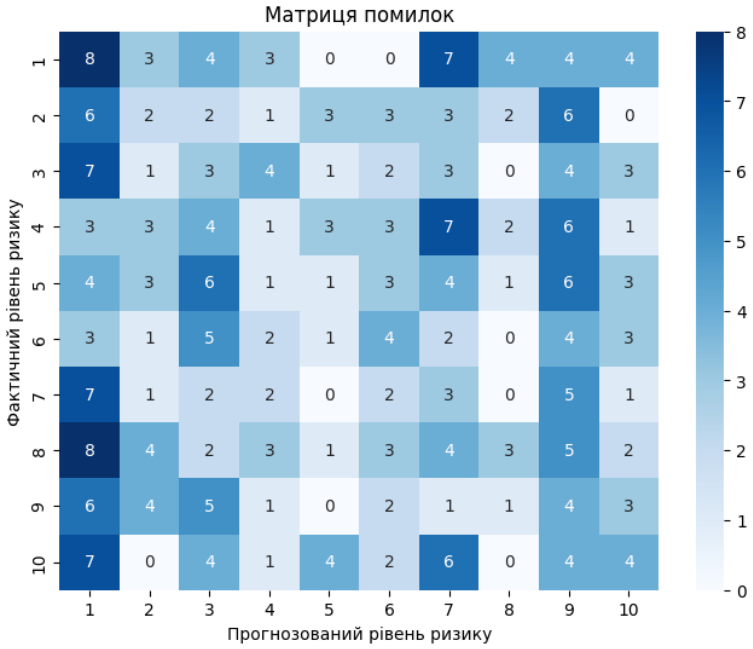


Рисунок 2.8 – Матриця помилок розподілку

Другий графік - Важливість факторів - відображає внесок кожної змінної у формування прогнозу. У цьому випадку можна побачити, що найбільший вплив на ризик мають такі фактори, як кількість змін у вимогах, перевищення бюджету та кількість критичних проблем. Це означає, що ці фактори мають найбільший вплив на ймовірність виникнення ризиків у проєкті, і їх необхідно моніторити в першу чергу.

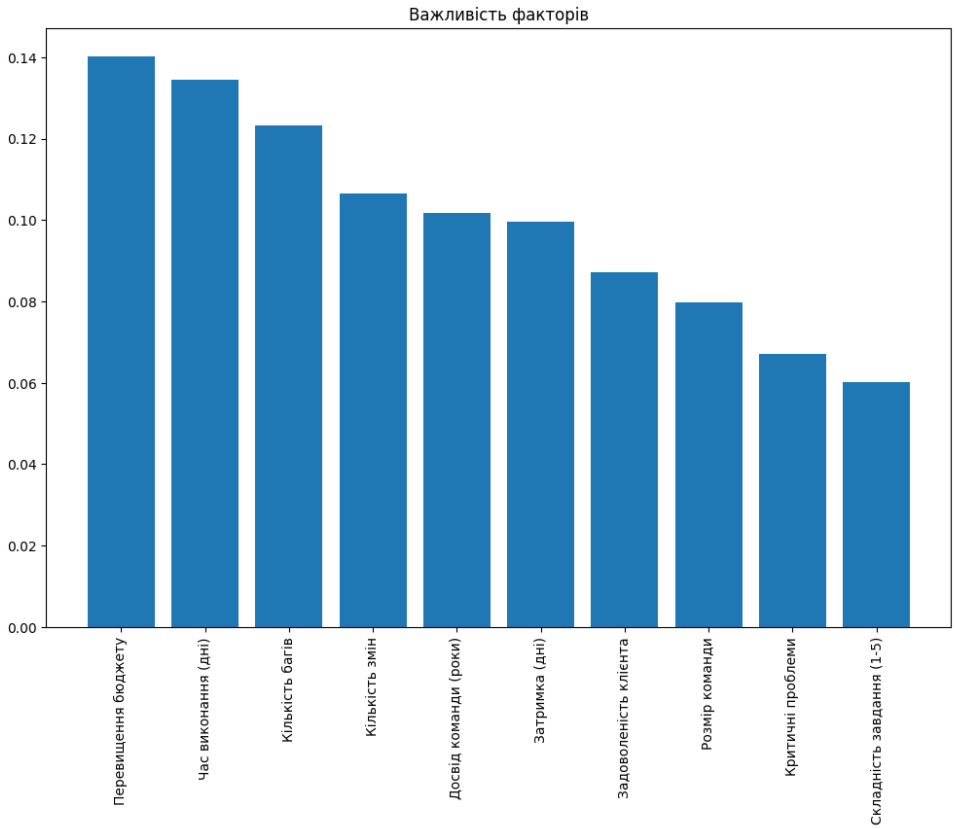


Рисунок 2.9 – Графік важливості факторів

Третій графік - ROC-крива - показує співвідношення між хибно-позитивною та істинно-позитивною частотами прогнозів. Чим ближче крива до верхнього лівого кута, тим точніше модель. AUC-значення, що відображається на графіку, демонструє загальну якість класифікації: чим вище AUC, тим краще модель справляється з розпізнаванням ризиків.

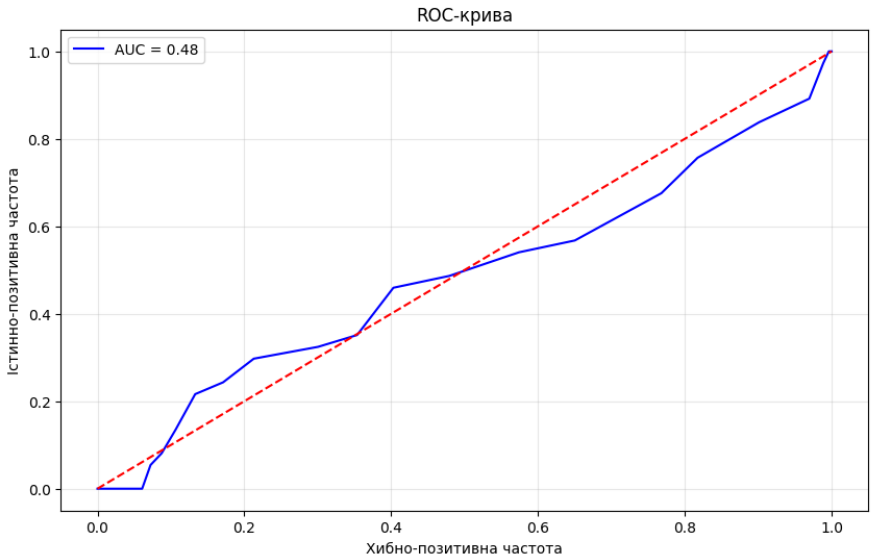


Рисунок 2.10 – ROC-крива методу Random Forest

Random Forest є одним із найефективніших алгоритмів для аналізу ризиків у проєктному менеджменті завдяки здатності працювати з великими обсягами даних і виявляти приховані закономірності. У контексті управління проєктами він дозволяє прогнозувати рівень ризику завдань на основі факторів, таких як кількість змін у вимогах, перевищення бюджету, кількість критичних проблем та затримки.

За рахунок ансамблю дерев рішень метод забезпечує високу точність прогнозування та знижує ризик перенавчання, що особливо важливо в умовах обмеженості даних та високої кореляції між змінними. Аналіз важливості факторів дозволяє не лише виявити ключові ризики, але й зрозуміти, які з них мають найбільший вплив на результат. Це сприяє обґрунтованому прийняттю управлінських рішень та розробці стратегій для зниження ризиків.

Таким чином, Random Forest демонструє високу ефективність як інструмент для виявлення та управління ризиками, що робить його цінним інструментом для проєктних менеджерів.

2.2.4 Градієнтний бустинг (XGBoost)

XGBoost - це потужний ансамблевий алгоритм машинного навчання, який використовує техніку градієнтного бустингу для побудови прогнозних моделей. На відміну від класичних дерев рішень, XGBoost створює серію дерев, кожне з яких навчається на залишках, тобто на помилках попереднього дерева. Цей підхід дозволяє алгоритму поступово коригувати прогноз, зменшуючи похибку на кожному кроці.

Процес починається зі створення початкової базової моделі, яка зазвичай є простим деревом, що прогнозує середнє значення цільової змінної. Потім обчислюються залишки - різниця між фактичними значеннями і прогнозами базової моделі. На основі цих залишків будується нове дерево, яке навчається саме на помилках попередньої моделі. Кожне наступне дерево додає свою корекцію до прогнозу, намагаючись мінімізувати залишки.

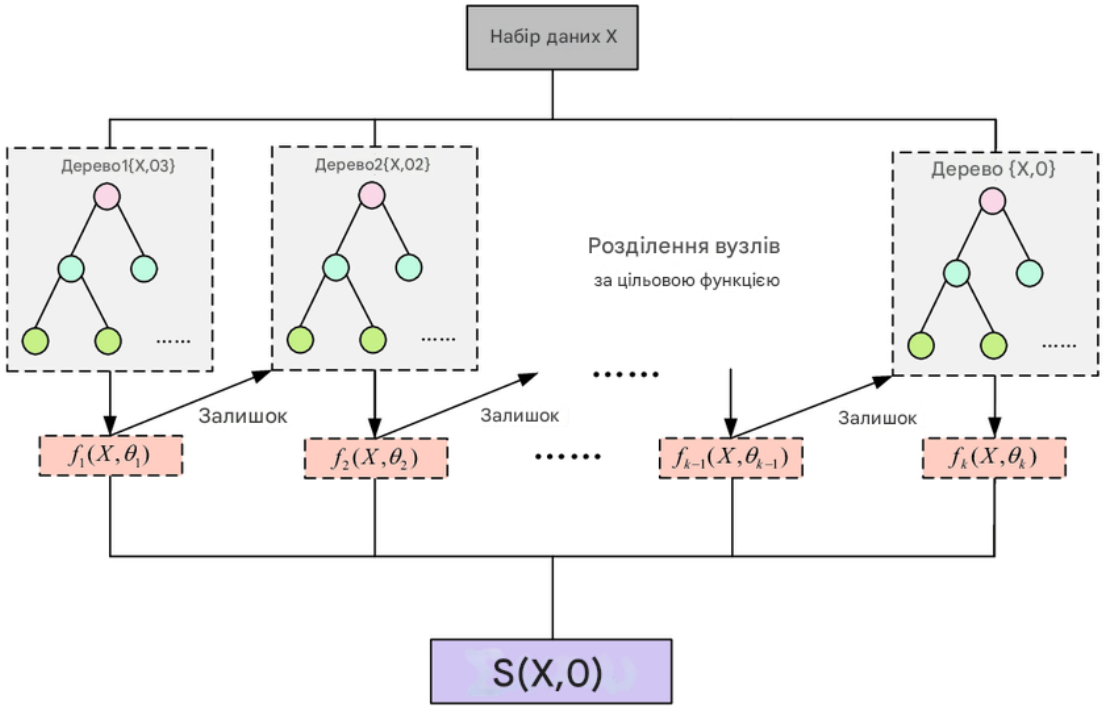


Рисунок 2.11 – Візуалізація методу XGboost

Для стабілізації навчання використовується параметр η - learning rate, який визначає, наскільки сильно кожне нове дерево впливає на остаточний прогноз. Це дозволяє алгоритму уникати перенавчання, особливо при великій кількості дерев. XGBoost також активно використовує регуляризацію - штраф за складність дерев. Це запобігає ситуації, коли модель надто точно підлаштовується під навчальні дані, але погано узагальнює нові дані.

На кожному етапі алгоритм обчислює функцію втрат, яка складається з двох компонентів: похибки прогнозу та штрафу за складність моделі. Завдяки цьому XGBoost може балансувати між точністю і простотою моделі, не втрачаючи здатності до узагальнення.

XGBoost є одним із найефективніших методів для ризик-менеджменту в проєктному менеджменті, оскільки здатен точно прогнозувати ризики на основі великої кількості факторів одночасно. У складних проєктах, де впливають такі фактори, як кількість змін у вимогах, перевищення бюджету, затримки у виконанні завдань та кількість критичних проблем, цей алгоритм дозволяє виявити фактори, які найбільше впливають на загальний рівень ризику.

XGBoost використовує ансамблевий підхід, де кожне дерево коригує помилки попереднього, формуючи більш точний прогноз. Це особливо корисно у випадках, коли проєкт має багато змінних із прихованими залежностями. Наприклад, аналіз одночасного впливу перевищення бюджету, кількості багів та затримок дозволяє алгоритму оцінити їхній сумарний вплив на ризики.

Основні переваги XGBoost у ризик-менеджменті:

- Алгоритм визначає фактори, які найбільше впливають на ризик, дозволяючи зосередитись на них;
- Стабільність прогнозів, вбудована регуляризація запобігає перенавчанню, забезпечуючи узагальнені прогнози;
- Аналіз складних залежностей, виявлення прихованих зв’язків між факторами, які неочевидні при традиційному аналізі.

• Обробка пропущених даних, здатність працювати навіть за відсутності частини даних, що є типовим для реальних проєктів.

Тобто XGBoost дозволяє не лише прогнозувати ризики, але й визначати зони з найбільшим впливом на проєкт, забезпечуючи своєчасне коригування плану дій для мінімізації можливих втрат.

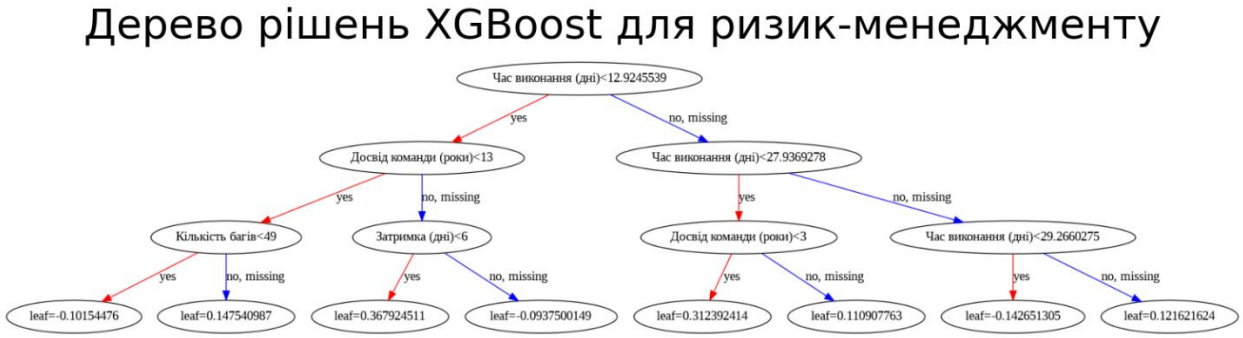


Рисунок 2.12 – Дерево рішень XGBoost для ризик-менеджменту

На прикладі зображення вище показано, як ми можемо використати алгоритм XGBoost у ризик-менеджменті для прогнозування рівня ризику на основі ключових факторів проєкту. Дерево рішень демонструє логіку прийняття рішень, починаючи з найбільш значущих змінних, таких як час виконання завдання, досвід команди та кількість багів.

Початковий вузол аналізує, чи є Час виконання (дні) меншим за 12.92. Якщо так, модель переходить до аналізу Досвід команди (роки), розділяючи дані на групи із досвідом меншим за 13 років та тими, що мають більший досвід. Далі йде перевірка на кількість багів, де визначається, чи перевищує їхня кількість 49. Таким чином, на кожному етапі модель уточнює прогноз, переходячи до наступної умови.

Якщо значення не відповідає жодній умові, модель переходить до листка (leaf), який містить прогноз ризику. Листи відображають підсумкове значення ризику для кожної гілки дерева. Наприклад, якщо Час виконання (дні) більше 29.26,

модель прогнозує ризик як 0.1216, вказуючи на низький ризик завершення завдання у встановлені терміни.

Таким чином, дерево рішень дозволяє менеджерам проєктів ідентифікувати найвпливовіші фактори, що сприяють підвищенню ризиків, і зрозуміти логіку їх взаємодії. Це надає можливість проактивно реагувати на зміни в проєкті, коригуючи план дій для мінімізації потенційних загроз.

Для прикладу, візьмемо ситуацію, коли необхідно оцінити ризики у проєкті розробки програмного забезпечення. Використовуючи набір характеристик, таких як кількість змін вимог, перевищення бюджету, кількість критичних помилок та затримок, формується прогноз рівня ризику проєкту за шкалою від 0 (низький ризик) до 7 (високий ризик).

Модель навчена. Звіт по тестовій вибірці:

	precision	recall	f1-score	support
0	0.88	0.81	0.84	47
1	0.57	0.49	0.53	47
2	0.45	0.60	0.51	47
3	0.33	0.35	0.34	46
4	0.37	0.34	0.36	47
5	0.45	0.43	0.44	47
6	0.54	0.57	0.56	47
7	0.84	0.81	0.83	47
accuracy			0.55	375
macro avg	0.56	0.55	0.55	375
weighted avg	0.56	0.55	0.55	375

==========

Рисунок 2.13 – Звіт по задачі на основі тестової вибірки

Основна користь такої оцінки полягає у виявленні найбільш критичних факторів ризику ще на етапі планування проєкту. Наприклад, проєкт із високим рівнем перевищення бюджету та великою кількістю змін вимог може отримати прогноз високого ризику. У такому випадку можна сформулювати рекомендації щодо зменшення кількості змін або жорсткішого контролю за бюджетом, що дозволить знизити ризики провалу проєкту.

Крім того, візуалізація основних факторів ризику на основі їх важливості допомагає ідентифікувати аспекти, які мають найбільший вплив на загальний ризик. Це дозволяє ефективніше розподілити ресурси та зосередитися на усуненні найбільш небезпечних ризиків.

```
============================================================
Введіть дані проекту для оцінки ризику (Ctrl+C для виходу):
Кількість змін вимог (0-25): 15
Середній досвід команди, роки (1-15): 5
Складність задачі (1-5): 3
Розмір команди (2-12): 7
Перевищення бюджету, % (0-100): 10
Очікувана тривалість, дні (5-45): 35
Очікувана задоволеність клієнта (1-10): 3
Очікувана кількість помилок (0-80): 50
Критичні проблеми (0-8): 6
Можлива затримка, дні (0-25): 15

--- Результати передбачення ---
Передбачений рівень ризику: 5 (0=низький, 7=високий)

Розподіл ймовірностей по рівнях ризику:
```

Рівень	Ймовірність
0	0.1%
1	1.9%
2	6.3%
3	16.8%
4	32.3%
5	36.0%
6	5.6%
7	0.9%

```
--- Рекомендації щодо зниження ризику ---
Зменшіть 'NumChanges' (значення: 15.0) для зниження ризику.
Зменшіть 'BudgetOverrunPct' (значення: 10.0) для зниження ризику.
Зменшіть 'CriticalIssues' (значення: 6.0) для зниження ризику.
Зменшіть 'DelayDays' (значення: 15.0) для зниження ризику.
```

Рисунок 2.14 – Відображення роботи тестової вибірки

2.3 Комбіноване використання алгоритмів машинного навчання в абстрактній системі управління проєктами

У реальних процесах керування проєктами виникають різнорідні дані: часові ряди активності команди, текстові описи завдань, ієрархічні зв'язки між етапами, спринтами й тасками. Жоден окремий алгоритм не охоплює всі аспекти цієї динаміки, тому практично доцільно застосовувати гібридні та ансамблеві стратегії:

- Stacking / Blending. LSTM прогнозує часові патерни (строки виконання, завантаження команди), Random Forest і XGBoost генерують оцінки складності та ризику; метамодель (наприклад, градієнтний бустинг) поєднує їхні виходи й формує інтегрований прогноз;
- Bagging із підпросторами ознак. Незалежні дерева Random Forest навчаються на різних підмножинах спринтових показників; результати агрегуються усередненням чи медіаною, знижуючи варіативність оцінок;
- Каскадні моделі. Швидкий регресійний блок (GBDT) відсіює завдання з низьким ризиком відставання; для «складних» залишків активується LSTM-аналіз історії змін, що дає кращий баланс швидкості й точності;
- Rule-based + ML. Поверх ансамблю встановлюється шар бізнес-правил (наприклад, ліміт 80 % завантаження кожного виконавця), забезпечуючи прозорість рішень і можливість оперативної корекції менеджером.

Таке комбінування підвищує точність прогнозів тривалості й вартості спринтів, робить систему стійкішою до шумних або неповних даних та забезпечує гнучку інтеграцію з будь-яким пайплайном DevOps - незалежно від конкретного інструментарію, що використовується в організації.

Реалізуємо програму на основі цих відомостей, вона буде мати наступний алгоритм:

- Ініціалізація середовища;

    - Перевіряє наявність потрібних Python-пакунків; відсутні встановлює через pip.
    - Створює службові каталоги /content/models та /content/data.
- Визначення схем даних;
    - Описує Pydantic-клас ProjectRecord, формалізуючи поля задачі, щоби одразу відсікти невалідні записи.
- Генератор синтетичного набору;
    - Псевдовипадково моделює 2 000 історій: ролі виконавців, story points, часові мітки комітів і збоїв тестів.
    - Серіалізує спискові поля у JSON-рядки й зберігає CSV.
- Завантаження та десеріалізація;
    - Читає CSV, повертає pandas.DataFrame; дати переводить у datetime, JSON-рядки - назад у списки.
- Інженерія ознак;
    - Обчислює статичні фічі (story points, blockers, one-hot ролей) та цілі:
        - delay_days - планова тривалість;
        - risk_flag - бінарна ознака можливого зриву.
    - Векторизує послідовності комітів/збоїв у 30-добовий LSTM-тензор.
- Навчання моделей;
    - LSTM → прогноз днів затримки.
    - Random Forest → оцінка складності (story points).
    - XGBoost → ймовірність зриву.
    - Gradient Boosting (стекінг) → інтегрує попередні прогнози для точнішого delay_days.
    - Фіксує OneHot-encoder і перелік фіч для подальшого інференсу; усе зберігає через joblib/keras.save().
- Оцінка ризику;

    - Агрегує три предиктори у компактний ризиковий бал; класифікує Low / Medium / High.
- Monte-Carlo сценарний аналіз;
    - Симулює 1 000 можливих дат завершення, припускаючи нормально розподілену невизначеність ±20 % навколо прогнозу.
- Інтерактивний CLI;
    - Відображає меню (Rich-таблиця) і приймає вибір через prompt_toolkit або input() fallback.
    - Пункти:
        - Завантажити / згенерувати дані.
        - Навчити моделі.
        - Прогноз спринта (затримка + складність + ризик).
        - Категорія ризику.
        - Monte-Carlo аналіз.
        - Експорт CSV-результатів.
        - Вихід.
- Узгодженість між train / predict;
    - При прогнозі відтворює порядок фіч і використовує той самий One-Hot-encoder; запобігає помилкам «feature names mismatch».
- Завершення.
    - Обробляє KeyboardInterrupt, коректно закриваючи сесію й залишаючи всі артефакти на диску.

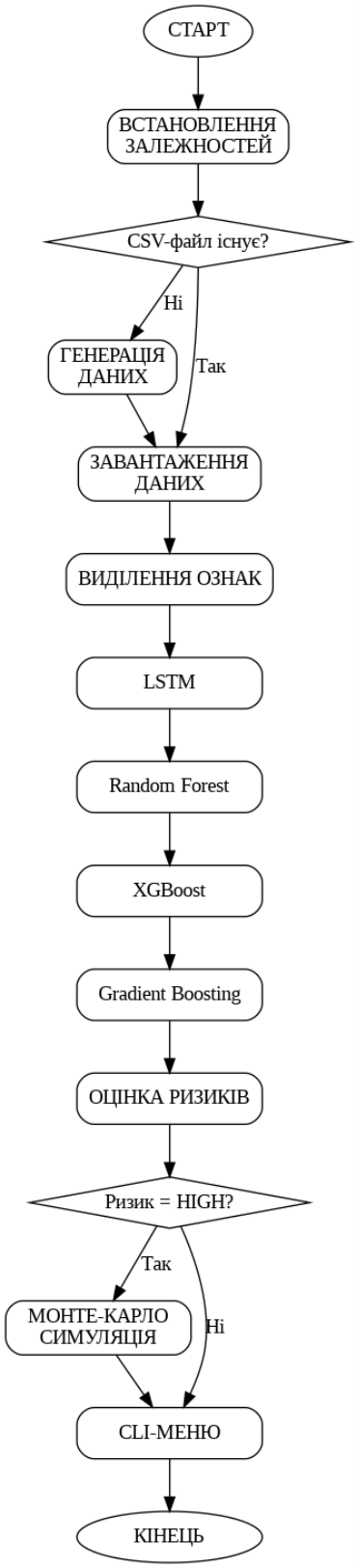


Рисунок 2.15 – Схема комбінованого використання алгоритмів

У рамках даної роботи розроблено та реалізовано програмний комплекс «ML-Project-Master», що виконує повний цикл аналітики життєвого циклу IT-завдань - від попередньої підготовки середовища до сценарного прогнозування кінцевих дат. Алгоритм ґрунтується на поетапному опрацюванні даних: спочатку автоматично встановлюються всі необхідні залежності, після чого система або завантажує наявний CSV-датасет, або генерує синтетичні приклади за відсутності таких даних. Далі здійснюється виділення статичних і часових ознак, що уможливлює побудову кількох незалежних моделей - LSTM-регресора для прогнозу затримки, Random Forest для оцінки складності, XGBoost для оцінки ймовірності зриву й комбінованого Gradient Boosting як метамоделі.

Результати кожного з предикторів об’єднуються у єдиний ризиковий показник, а в разі категорії HIGH автоматично ініціюється модуль Монте-Карло, котрий моделює тисячу потенційних сценаріїв завершення задачі. Завдяки цьому менеджер проєкту отримує не лише точкові оцінки, а й розподіл імовірних дат та коридор довіри, що значною мірою підвищує обґрунтованість ухвалених рішень.

Управління всіма можливостями здійснюється через інтерактивне CLI-меню: користувач послідовно обирає дії - від генерації даних до експорту результуючих прогнозів. Інтерфейс побудовано із застосуванням бібліотеки Rich, тож інформація подається у зручних таблицях, панелях і прогрес-барах, а fallback-механізм гарантує коректну роботу навіть у середовищах без підтримки prompt_toolkit.

Показана у роботі блок-схема, укладена за вимогами ГОСТ, відображає ключові вузли алгоритму та його розгалуження: перевірку наявності даних і відгалуження процесу в залежності від критичності ризику. Це чітко ілюструє логіку переходів, забезпечуючи прозорість та відтворюваність усіх кроків.

Далі буде продемонстровано роботу алгоритму на прикладі реального кейсу з портфеля продуктових задач: послідовно покажемо підготовку даних, навчання моделей, формування ризикових інсайтів і сценарний аналіз із використанням симуляції Монте-Карло. Порівняння фактичних і прогнозних даних доведе

практичну ефективність підходу та наочно підтвердить його застосовність у щоденній діяльності проджект-менеджера.

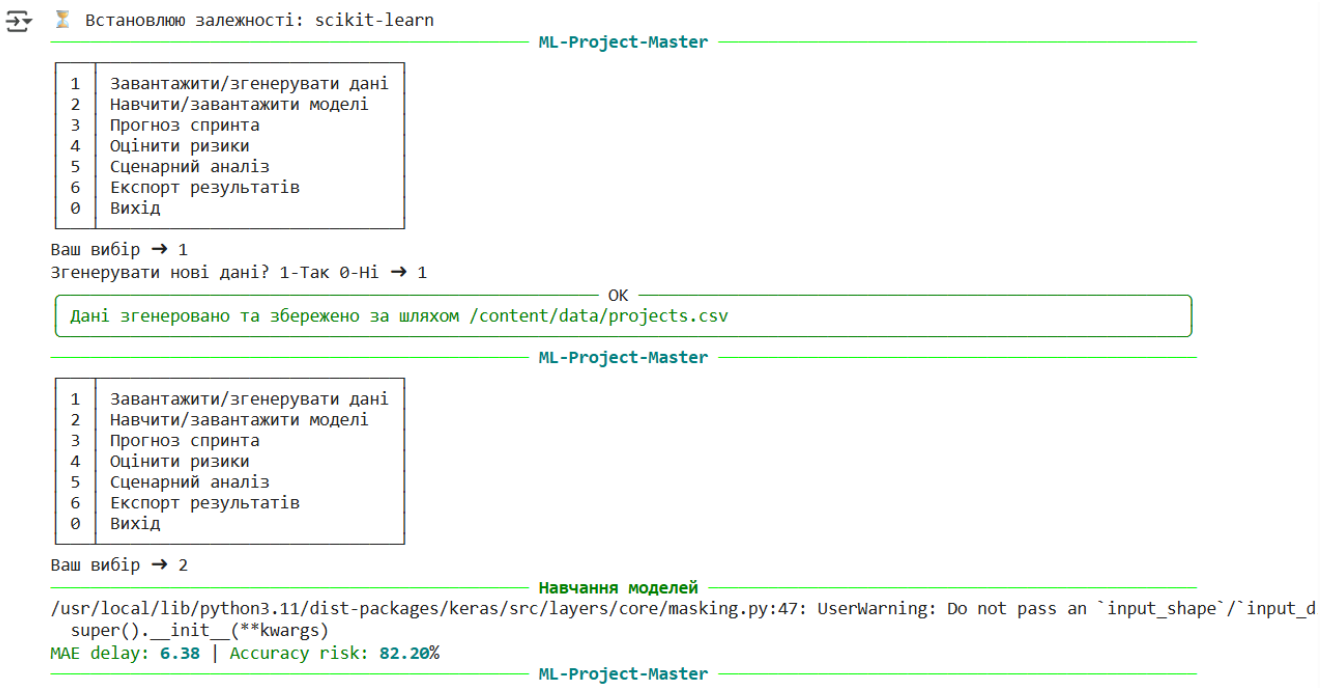


Рисунок 2.16 – Скріншот виконання програми ML-Project-Master

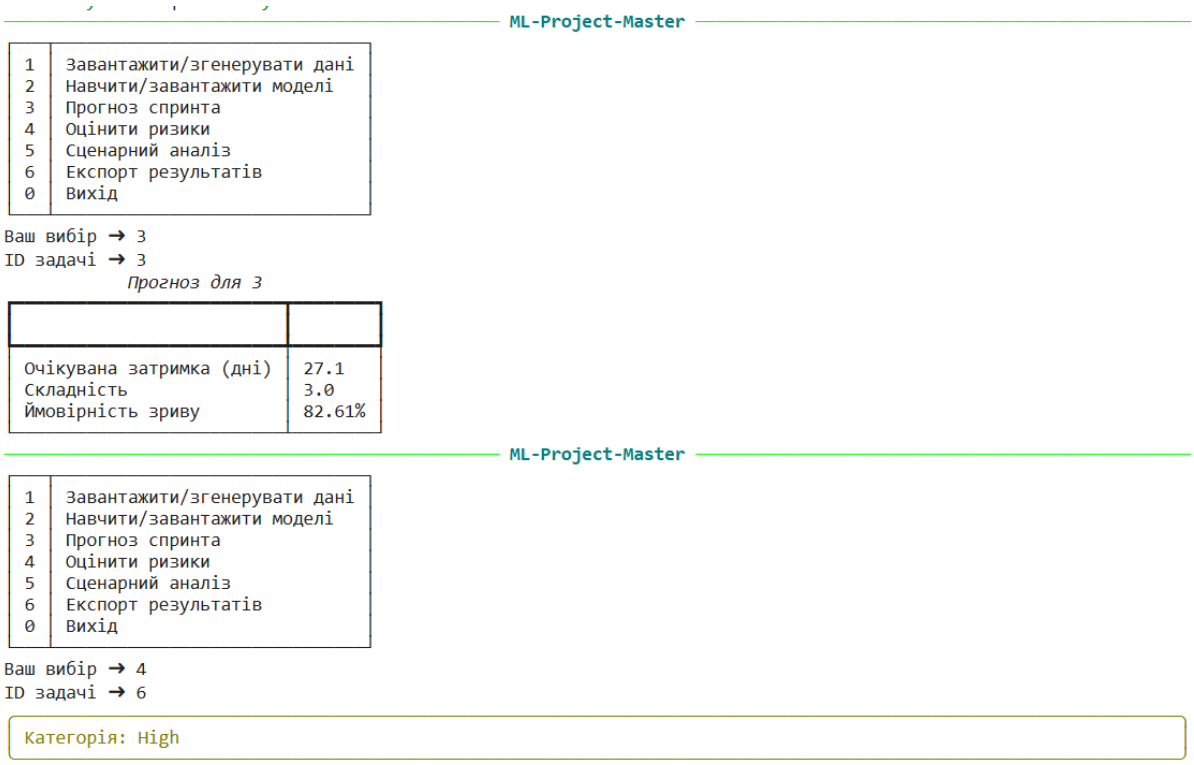


Рисунок 2.18 – Скріншот виконання програми ML-Project-Master

```
| 1 | Завантажити/згенерувати дані
| 2 | Навчити/завантажити моделі
| 3 | Прогноз спринта
| 4 | Оцінити ризики
| 5 | Сценарний аналіз
| 6 | Експорт результатів
| 0 | Вихід

Ваш вибір ➜ 5
ID задачі ➜ 2

Monte Carlo
Раніш: 2025-03-27
Ймовірно: 2025-04-07
Пізно: 2025-04-19

ML-Project-Master

| 1 | Завантажити/згенерувати дані
| 2 | Навчити/завантажити моделі
| 3 | Прогноз спринта
| 4 | Оцінити ризики
| 5 | Сценарний аналіз
| 6 | Експорт результатів
| 0 | Вихід

Ваш вибір ➜ 6

Експортовано ➜ /content/results_export.csv
```

Рисунок 2.19 – Скріншот виконання програми ML-Project-Master

results_export.csv

1 to 10 of 2000 entries Filter

id	summary	story_points	assignee_role	created_at	due_date	state_history	commits_ts	test_failures_ts	blockers
1	Задача 1	5	Frontend	2025-05-13 14:06:38.118753	2025-06-13 14:06:38.118753	['Code Review', 'To Do', 'In Progress', 'Testing', 'Done']	[datetime.datetime(2025, 5, 26, 14, 6, 38, 118753), datetime.datetime(2025, 6, 8, 14, 6, 38, 118753), datetime.datetime(2025, 5, 15, 14, 6, 38, 118753), datetime.datetime(2025, 6, 3, 14, 6, 38, 118753), datetime.datetime(2025, 5, 19, 14, 6, 38, 118753), datetime.datetime(2025, 5, 15, 14, 6, 38, 118753), datetime.datetime(2025, 5, 29, 14, 6, 38, 118753), datetime.datetime(2025, 6, 12, 14, 6, 38, 118753), datetime.datetime(2025, 6, 4, 14, 6, 38, 118753)]	[datetime.datetime(2025, 5, 28, 0, 6, 38, 118753), datetime.datetime(2025, 6, 10, 3, 6, 38, 118753), datetime.datetime(2025, 5, 16, 15, 6, 38, 118753)]	2
2	Задача 2	3	Frontend	2025-03-20 14:06:38.118753	2025-04-08 14:06:38.118753	['To Do', 'In Progress', 'Done', 'Testing', 'Code Review']	[datetime.datetime(2025, 3, 21, 14, 6, 38, 118753), datetime.datetime(2025, 3, 30, 14, 6, 38, 118753), datetime.datetime(2025, 4, 5, 14, 6, 38, 118753), datetime.datetime(2025, 3, 21, 14, 6, 38, 118753), datetime.datetime(2025, 4, 5, 14, 6, 38, 118753), datetime.datetime(2025, 4, 4, 14, 6, 38, 118753)]	[datetime.datetime(2025, 3, 22, 20, 6, 38, 118753)]	2
3	Задача 3	3	Backend	2025-04-30 14:06:38.118753	2025-05-16 14:06:38.118753	['To Do', 'In Progress', 'Testing']	[datetime.datetime(2025, 4, 30, 14, 6, 38, 118753), datetime.datetime(2025, 5, 8, 14, 6, 38, 118753), datetime.datetime(2025, 5, 2, 14, 6, 38, 118753), datetime.datetime(2025, 5, 11, 14, 6, 38, 118753), datetime.datetime(2025, 5, 10, 14, 6, 38, 118753), datetime.datetime(2025, 5, 14, 14, 6, 38, 118753), datetime.datetime(2025, 5, 11, 14, 6, 38, 118753), datetime.datetime(2025, 5, 5, 14, 6, 38, 118753), datetime.datetime(2025, 5, 15, 14, 6, 38, 118753),	[datetime.datetime(2025, 5, 2, 9, 6, 38, 118753)]	2

Рисунок 2.20 – Таблиця яка є результатом виконання програми

2.3 Проєктування архітектури системи автоматизації управління IT-проєктами на основі методів машинного навчання

Проєктування архітектури системи автоматизації управління IT-проєктами базується на принципах модульності, масштабованості та можливості інтеграції з існуючими інформаційними середовищами управління проєктами. Метою архітектури є створення інтелектуального програмного рішення, яке здатне аналізувати історичні дані про виконання завдань, виявляти закономірності, формувати прогнози тривалості робіт, оцінювати ризики відхилення від плану та пропонувати управлінські рекомендації. Основна ідея полягає у впровадженні машинного навчання як інструмента підтримки прийняття рішень для керівників проєктів.

Система реалізується як набір взаємопов'язаних модулів, кожен із яких виконує окрему функцію в загальному процесі обробки даних та побудови прогнозів. Загальна архітектура має вигляд послідовного конвеєра, де кожен етап формує вхід для наступного. У спрощеній формі структура системи описується таким алгоритмом:

1. Збір даних - імпорт історичних даних про проєкти, завдання, строки, виконавців і результати спринтів у форматі CSV або JSON (експорти з Jira, Trello, Asana);
2. Попередня обробка - очищення, нормалізація, видалення пропусків, перетворення дат, кодування категоріальних змінних, перевірка цілісності даних;
3. Формування ознак (feature engineering) - створення нових показників на основі вхідних даних: середня тривалість виконання задач певного типу, кількість блокерів, середня продуктивність виконавця, щільність завантаження спринтів;
4. Навчання моделей машинного навчання - тренування кількох алгоритмів для різних аналітичних задач:

- Random Forest або Gradient Boosting - прогноз тривалості виконання задач (регресія);
- Logistic Regression або XGBoost - оцінка ризику зриву дедлайну (класифікація);
- ARIMA або LSTM - прогнозування темпів виконання задач у часі (аналіз часових рядів).

5. Оцінювання результатів - порівняння прогнозів із реальними даними, обчислення метрик точності (MAE, RMSE, ROC AUC, MAPE), вибір найкращої моделі;
6. Візуалізація та інтерфейс користувача - створення аналітичної панелі на базі бібліотеки Streamlit, де користувач може завантажити свої дані, отримати прогнозні показники, переглянути графіки продуктивності, burndown-чарти та ризикові індикатори;
7. Збереження результатів - експорт прогнозів у таблиці, формування управлінських звітів, збереження моделей у форматі .pkl або .joblib для подальшого використання.

Реалізація системи виконується мовою програмування Python, що забезпечує поєднання інструментів для обробки даних, побудови моделей і розробки простого веб-інтерфейсу. Використовуються такі основні бібліотеки:

- pandas - обробка та аналіз табличних даних;
- scikit-learn - побудова моделей машинного навчання;
- xgboost - реалізація градієнтного бустингу;
- statsmodels або prophet - аналіз часових рядів;
- matplotlib та seaborn - візуалізація даних і результатів моделювання;
- streamlit - розробка інтерактивного інтерфейсу користувача.

Результатом проєктування стане прототип програмного продукту (Python-застосунок), який забезпечує автоматизоване прогнозування ключових показників проєктного циклу: тривалості виконання задач, імовірності зриву строків і динаміки

виконання робіт. Такий підхід демонструє практичну можливість використання алгоритмів машинного навчання для підвищення обґрунтованості управлінських рішень, мінімізації ризиків і підвищення ефективності планування в IT-проєктах.

2.3.1 Архітектурна структура та блок-схема функціонування системи

Архітектура розроблюваної системи автоматизації управління IT-проєктами є модульною та побудована за принципом послідовного оброблення даних. Такий підхід дозволяє розділити процес роботи системи на логічні рівні, кожен з яких виконує власну функцію, зберігаючи при цьому єдину цілісну логіку. Основною ідеєю є забезпечення безперервного потоку даних - від моменту їх отримання до генерації прогнозів і формування управлінських звітів.

На початковому рівні система отримує дані у форматі CSV або JSON, експортовані з інструментів управління проєктами, таких як Jira, Trello чи Asana. Ці дані містять інформацію про завдання, виконавців, спринти, строки, статуси та показники продуктивності команди. Вхідний потік проходить через модуль збору та стандартизації, де відбувається перевірка структури, уніфікація форматів, нормалізація назв і перетворення даних до єдиного виду. Це необхідно для подальшої обробки, оскільки джерела мають різну внутрішню логіку опису процесів.

Далі дані передаються у модуль попередньої обробки та формування ознак. Тут здійснюється очищення від пропусків, виправлення аномалій, переведення категоріальних змінних у числову форму, а також створення похідних параметрів, які впливають на прогноз. На цьому етапі формується набір ознак, що описує кожну задачу у вигляді числового вектора. Зокрема, враховується середня тривалість подібних завдань, кількість блокерів, стабільність виконавця, показники попередніх спринтів, кількість переоцінок та швидкість закриття задач. Після обробки формується єдиний файл з узгодженими полями, який використовується для побудови моделей машинного навчання.

Ключовим елементом архітектури є модуль моделювання та прогнозування. На цьому етапі реалізується логіка машинного навчання. Для прогнозування тривалості виконання завдань застосовуються регресійні ансамблеві моделі, такі як Random Forest або XGBoost. Для оцінювання ризику зриву строків використовується класифікаційний підхід, де модель на основі історичних характеристик спринтів визначає ймовірність невиконання плану. Для прогнозування продуктивності команди в часі використовуються моделі часових рядів, зокрема ARIMA або нейронні мережі типу LSTM. Результати кожної моделі зберігаються у вигляді окремих файлів і можуть бути повторно використані без необхідності повторного навчання.

Після отримання прогнозів активується модуль оцінювання та аналітики. У цьому модулі проводиться обчислення показників якості, таких як середня абсолютна похибка, корінь середньоквадратичної похибки, точність класифікації та відсоток похибки прогнозу. Візуалізація метрик і ваг ознак дозволяє оцінити внесок кожного параметра у фінальне рішення моделі, що підвищує її прозорість і придатність для управлінського використання.

Завершальним елементом архітектури є користувацький інтерфейс, реалізований за допомогою бібліотеки Streamlit. У ньому користувач має можливість завантажити власні дані, виконати навчання моделей, переглянути прогнозні результати у вигляді таблиць і графіків, а також експортувати звіти у форматі CSV або PNG. Інтерфейс реалізує три основні функціональні блоки - прогноз тривалості завдань, оцінку ризику зриву спринту та прогноз темпів виконання робіт. Усі елементи взаємодіють у межах єдиного обчислювального конвеєра, де кожен рівень автоматично передає результати наступному.

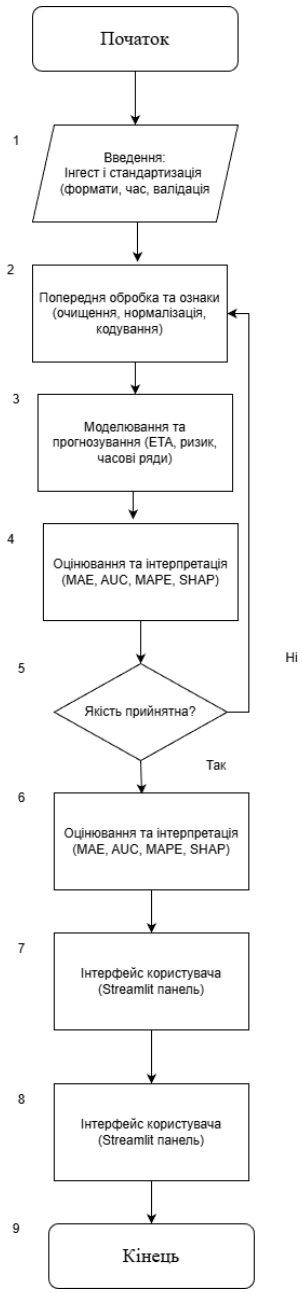


Рисунок 2.21 – Блок-схема структури функціонування системи

Така структура забезпечує чітку послідовність обробки інформації, дає можливість відокремлено розвивати окремі компоненти системи, а також гарантує стабільність і передбачуваність результатів. У підсумку архітектура поєднує аналітичну точність алгоритмів машинного навчання з практичною корисністю

інструментів управління проєктами, формуючи інтелектуальну основу для ухвалення обґрунтованих рішень у процесі керування ІТ-проєктами.

2.3.2 Логічна структура та принцип роботи веб-системи

Система розробляється у вигляді веб-застосунку, що поєднує алгоритми машинного навчання з простим інтерфейсом користувача. Її основна мета - надати можливість керівнику проєкту виконувати аналітичні операції (оцінку ризиків, пріоритизацію задач, прогнозування завантаженості) через зручну веб-панель без потреби у складному програмному середовищі.

Функціонування системи побудовано за принципом клієнт–серверної архітектури. Серверна частина, реалізована мовою Python, виконує аналітичні обчислення, навчання моделей і формування прогнозів. Клієнтська частина відповідає за візуалізацію результатів і взаємодію користувача із системою через вебінтерфейс, створений засобами бібліотеки Streamlit.

Робота веб-застосунку організована послідовно. Користувач обирає тип аналізу (ризики, продуктивність, завантаженість персоналу), після чого система звертається до відповідної моделі машинного навчання, виконує обчислення та виводить результат у вигляді таблиць, графіків і коротких висновків. Моделі зберігаються на сервері у вигляді серіалізованих файлів і можуть бути оновлені при повторному навчанні.

Таблиця 2.3 Взаємодія між складовими системи

Компонент системи	Основна функція	Реалізація
Інтерфейс користувача	Вибір типу аналізу, запуск моделі, перегляд результатів	Streamlit (Python UI)
Аналітичне ядро	Обробка даних, виклик моделей, формування прогнозів	Python, scikit-learn, TensorFlow

База моделей	Зберігання навчених моделей та гіперпараметрів	Формат .joblib / .h5
Модуль візуалізації	Побудова графіків, таблиць і текстових звітів	Matplotlib, Plotly
Серверна частина	Обробка запитів, логіка маршрутизації	Python runtime (хмарний)

Така структура забезпечує зрозумілий цикл взаємодії: користувач → аналітичне ядро → прогноз → візуалізація. У результаті веб-застосунок виступає не лише інструментом аналізу, а й інтерактивним середовищем підтримки прийняття управлінських рішень у межах проєктної діяльності.

2.3.3 Вибір і налаштування алгоритмів машинного навчання для задач системи

Для реалізації системи автоматизації управління проєктами було обрано набір алгоритмів машинного навчання, які забезпечують вирішення різних аналітичних задач - оцінку ризиків, пріоритизацію завдань, прогнозування завантаженості персоналу та виявлення потенційних проблем. Основним критерієм вибору слугувала здатність моделей працювати з невеликими вибірками, швидко адаптуватися до оновлених даних і забезпечувати високу точність прогнозів за обмеженого обсягу інформації.

Алгоритми поділяються на три основні групи: регресійні, класифікаційні та моделі часових рядів. Регресійні моделі застосовуються для кількісних прогнозів, таких як визначення очікуваної тривалості виконання задач. Класифікаційні моделі використовуються для оцінювання ймовірності виникнення ризиків чи проблем. Моделі часових рядів, зокрема рекурентні нейронні мережі, дозволяють аналізувати динаміку змін у завантаженості команди та виявляти тенденції.

Підбір алгоритмів здійснювався експериментально на основі порівняння показників точності, стабільності та швидкодії. Для регресійних задач обрано Random Forest Regressor, оскільки він поєднує високу точність і стійкість до шумів у даних. Для класифікації ризиків ефективно зарекомендував себе XGBoost Classifier, який забезпечує кращу збіжність і контроль над переобученням. Для аналізу часових залежностей використано LSTM (Long Short-Term Memory) - архітектуру нейронної мережі, здатну запам'ятовувати попередні стани системи й робити прогнози на основі історії подій.

Налаштування моделей виконувалося у середовищі Python із використанням бібліотек scikit-learn та TensorFlow. Підбір гіперпараметрів здійснювався за допомогою методів перехресної перевірки (GridSearchCV) із розділенням вибірки на навчальну, валідаційну та тестову частини у співвідношенні 70:20:10. Для контролю якості застосовувалися такі метрики: MAE і RMSE для регресійних моделей, ROC AUC і Accuracy для класифікаційних, MAPE для часових рядів.

Отримані результати свідчать про високу ефективність ансамблевих методів порівняно з базовими алгоритмами лінійної регресії чи логістичної класифікації. Впровадження нейронної мережі LSTM дало змогу підвищити точність прогнозу завантаженості команди на 10–15% завдяки здатності враховувати довготривалі залежності між етапами проєкту.

3 РЕАЛІЗАЦІЯ ТА ЕКСПИРЕМЕНТАЛЬНІ РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1 Прототипування інтелектуальної системи автоматизації управління проєктами

Прототипування системи виконано з метою перевірки логіки роботи запропонованої моделі та візуалізації принципів взаємодії користувача з аналітичними модулями. На цьому етапі не передбачалося повне програмне впровадження алгоритмів, а лише побудова структурної моделі інтерфейсу, що демонструє основні функції та послідовність дій користувача.

Система розглядається як веб-застосунок із мінімальним набором елементів, необхідних для виконання базових аналітичних завдань. Користувач отримує доступ до єдиної панелі, де може переглядати дані, ініціювати аналітичні обчислення та переглядати результати прогнозів у зручному форматі. Інтерфейс побудовано за принципом простоти та інтуїтивності, що забезпечує швидке сприйняття інформації без перевантаження деталями.

Прототип складається з кількох логічних екранів, які відповідають основним етапам роботи користувача. На головному екрані відображається коротка інформація про стан проєкту та кнопки переходу до модулів. Далі користувач може обрати тип аналізу: оцінку ризиків, прогнозування завантаженості або пріоритизацію завдань. Після вибору система моделює результат у вигляді таблиць і простих графіків, які ілюструють тенденції та потенційні проблеми.

Таблиця 3.1 Опис необхідних UI елементів для системи

Елемент інтерфейсу	**Призначення**	**Тип відображення**
Головна сторінка	Навігація між модулями, коротка статистика проєкту	Панель з індикаторами та кнопками

Модуль “Ризики”	Перегляд ймовірності затримок або перевантажень	Таблиця з рівнями ризику
Модуль “Завантаженість”	Оцінка розподілу робочого часу та ресурсів	Проста гістограма або лінійний графік
Модуль “Пріоритизація”	Формування черговості завдань за важливістю	Список із маркуванням пріоритетів
Вікно результатів	Підсумкове відображення прогнозів і коротких висновків	Таблиця з описом та графік тренду

Прототипування в даному дослідженні розглядається як метод наукового моделювання, що дозволяє перевірити концептуальні рішення до початку повномасштабної розробки системи. Його застосування ґрунтується на принципах системного підходу, згідно з яким складна інформаційна система спочатку подається у вигляді спрощеної моделі з чітко визначеними функціями, зв’язками та поведінковими характеристиками.

Метод прототипування забезпечує можливість поступового уточнення вимог, перевірки логіки роботи користувача й узгодження структури майбутнього інтерфейсу з аналітичними цілями системи. У процесі побудови прототипу реалізується принцип ітераційності: кожен етап передбачає аналіз, уточнення та вдосконалення попередньої версії з урахуванням результатів спостережень.

Використання прототипування науково обґрунтоване тим, що воно дозволяє здійснювати верифікацію гіпотез про зручність, зрозумілість і ефективність подання інформації без необхідності програмної реалізації всіх функцій. Такий підхід зменшує ризики невідповідності між проектною моделлю та очікуваною поведінкою системи.

Отже, прототипування виступає не лише технічним інструментом, а й дослідницьким методом, який поєднує елементи когнітивного аналізу, системного

проектування та візуального моделювання, що дозволяє сформувати науково перевірену основу для подальшої реалізації аналітичної системи управління проєктами.

Таким чином, прототипування дозволило відтворити спрощену модель майбутньої системи, оцінити її структуру, логіку та зручність використання. Отримана модель підтвердила доцільність побудови аналітичного інтерфейсу з мінімалістичним набором елементів, який забезпечує швидкий доступ до ключової інформації та може слугувати основою для подальшої розробки програмної системи.

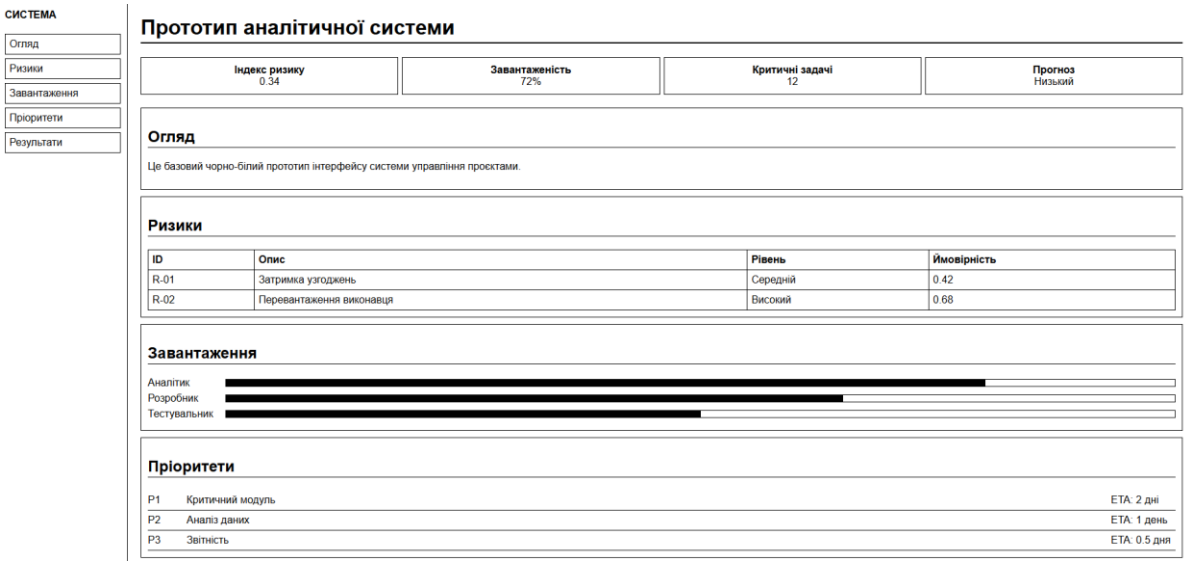


Рисунок 3.1 – Прототип інтерфейсу

Прототип системи є спрощеним чорно-білим макетом інтерфейсу, створеним для візуалізації логіки роботи та перевірки структури майбутнього застосунку. Він не виконує аналітичних обчислень, а лише демонструє послідовність дій користувача й базову організацію даних.

Основні елементи прототипу:

- Бічне меню - забезпечує навігацію між основними розділами;
- Панель показників (KPI) - відображає ключові узагальнені метрики стану проєкту;

- Модуль “Ризики” - таблиця з прикладами потенційних проблем і рівнями ризику;
- Модуль “Завантаження” - лінійні індикатори навантаження учасників команди;
- Модуль “Пріоритети” - спрощений список задач із позначенням черговості виконання;
- Блок “Результати” - текстовий підсумок стану проєкту.

Такий прототип дозволяє оцінити логіку взаємодії, читабельність і розташування елементів інтерфейсу до початку технічної реалізації.

Вибір веб-інтерфейсу як основного формату для прототипування системи зумовлений його універсальністю, доступністю та наочністю в умовах дослідницького середовища. Веб-застосунок дозволяє відтворити типову взаємодію користувача із системою управління проєктами без необхідності встановлення додаткового програмного забезпечення, що є важливим для демонстрації функціональної моделі у навчальному чи науковому контексті.

Формат веб-інтерфейсу забезпечує незалежність від операційної системи, сумісність із більшістю пристроїв і простоту поширення результатів дослідження. Його структура природно відображає логіку управлінського процесу - переходи між розділами, перегляд показників і формування висновків. Це дозволяє оцінити не лише аналітичну, а й когнітивну ефективність майбутньої системи.

Крім того, веб-інтерфейс зручний для подальшої інтеграції з модульною архітектурою аналітичного ядра, оскільки передбачає стандартизований обмін даними через HTTP-запити. Такий формат створює можливість для розширення прототипу до повноцінного веб-застосунку, здатного обробляти результати роботи алгоритмів машинного навчання в реальному часі.

Функціональні сценарії відображають типові дії користувача під час взаємодії з прототипом системи та демонструють логіку проходження інформаційних

процесів. Вони розроблені для оцінки послідовності аналітичних етапів, зрозумілості навігації й ефективності візуального представлення результатів.

Прототип імітує роботу системи управління проєктами у спрощеному вигляді, де користувач може виконувати базові операції без залучення складних аналітичних обчислень. Кожен сценарій відповідає окремому завданню управління: моніторингу стану проєкту, аналізу ризиків, оцінюванню завантаженості персоналу чи визначенню пріоритетності задач.

Основні функціональні сценарії:

1. Огляд проєкту. Користувач відкриває головну панель, де подано узагальнені показники - індекс ризику, рівень завантаження команди, кількість критичних завдань і прогнозований стан проєкту;
2. Оцінка ризиків. У модулі “Ризики” відображається таблиця з умовними прикладами можливих відхилень. Користувач переглядає рівень ризику й отримує короткі текстові рекомендації щодо дій;
3. Аналіз завантаженості. На сторінці “Завантаження” користувач бачить прості лінійні індикатори, які ілюструють розподіл робочого навантаження між учасниками команди;
4. Пріоритизація завдань. У модулі “Пріоритети” наведено список задач із маркуванням за рівнем важливості (P1–P3) та орієнтовним часом виконання;
5. Отримання результатів. Після проходження всіх етапів користувач переглядає узагальнений текстовий висновок, який демонструє роботу прототипу як цілісної моделі прийняття управлінських рішень.

Такі сценарії забезпечують перевірку логічної структури системи, взаємозв’язку між її елементами та відповідності розробленого інтерфейсу принципам аналітичної послідовності “сприйняття даних → аналіз → висновок”.

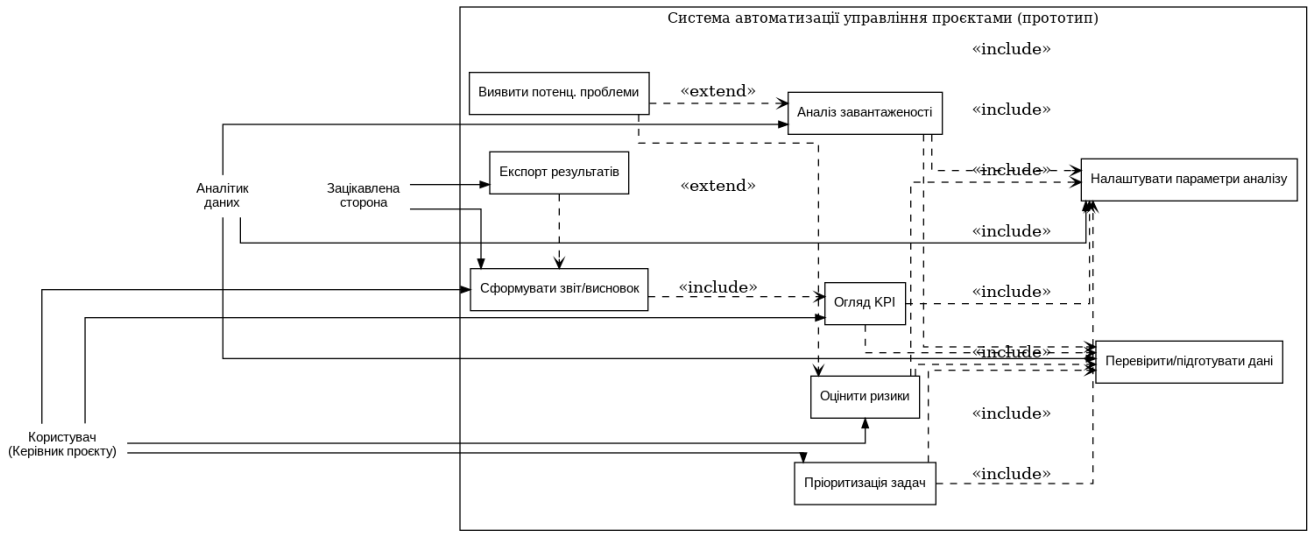


Рисунок 3.2 – Use-Case діаграма можливого використання системи

Основні учасники процесу:

- Користувач (керівник проєкту) - ініціює основні аналітичні дії, керує процесом оцінювання ризиків, пріоритизації задач і формування підсумкових звітів;
- Аналітик даних - відповідає за підготовку вхідних даних, налаштування параметрів аналізу та перевірку результатів обчислень;
- Зацікавлена сторона - отримує узагальнені звіти та результати експорту для прийняття управлінських рішень.

Основні варіанти використання системи:

- Перевірити/підготувати дані та налаштувати параметри аналізу - підготовчі етапи, що забезпечують коректність вхідної інформації;
- Огляд KPI, оцінка ризиків, аналіз завантаженості та пріоритизація задач - основні аналітичні функції системи, які формують інформаційну основу для прийняття рішень;
- Виявлення потенційних проблем - розширений аналітичний етап, який базується на попередніх розрахунках ризиків і навантаження;

- Формування звіту та експорт результатів - завершальні етапи, що забезпечують створення та передачу узагальнених висновків.

Логіка зв’язків:

Взаємозв’язки між блоками реалізовані через відношення «include» і «extend». Основні аналітичні процеси *включають* (include) підготовку даних і налаштування параметрів. Модуль “Виявлення проблем” *розширює* (extend) функції ризик-аналізу та оцінювання завантаженості. Формування звіту включає огляд ключових показників, а експорт результатів базується на вже сформованому звіті.

Таким чином, діаграма демонструє цілісну логіку роботи системи - від підготовки даних і аналізу показників до формування підсумкових висновків, відображаючи роль кожного користувача у цьому процесі та взаємозалежність аналітичних модулів.

3.2 Використання хмарних обчислювальних технологій для експериментальної реалізації системи

3.2.1 Використання хмарних обчислювальних технологій (Google Colab) для реалізації та тестування моделей

У межах експериментальної частини дослідження для навчання, тестування та аналізу моделей машинного навчання застосовано хмарне середовище Google Colab, яке забезпечує повноцінну інфраструктуру для обчислень без потреби у локальному високопродуктивному обладнанні. Використання хмарної платформи дозволяє розгорнути відтворювані експерименти, стандартизувати середовище виконання та уникнути апаратних обмежень персональних комп'ютерів.

Хмарна інфраструктура Colab забезпечує автоматичне підключення до апаратних прискорювачів (GPU або TPU), що дає змогу суттєво скоротити час тренування складних моделей (LSTM, GRU, ансамблевих методів). Система динамічно призначає ресурси та гарантує стабільність виконання, уникаючи впливу локальних системних параметрів чи конфігурацій.

Вбудована інтеграція з Google Drive забезпечує централізоване керування наборами даних, журналами експериментів, проміжними моделями та артефактами. Це створює чітку структуру зберігання та спрощує повторне використання попередніх результатів. Середовище дозволяє документувати робочий процес у вигляді інтерактивних комірок, де код, графіки та коментарі поєднані в єдиному дослідницькому сценарії.

Для забезпечення контролю якості процесу навчання моделі, фіксації проміжних результатів та візуалізації було використано стандартні бібліотеки Python (Matplotlib, Seaborn, Plotly). Візуальні графіки та журнали дозволяють аналізувати динаміку навчання, виявляти перенавчання, а також порівнювати різні конфігурації алгоритмів.

Використання Google Colab забезпечило такі переваги: стандартизоване середовище виконання, можливість швидкого масштабування, інтерактивність,

зручність документування, відтворюваність експериментів та незалежність від локальної апаратної бази.

Таблиця 3.2 - Використані хмарні ресурси Google Colab

Компонент	Призначення	Характеристика
GPU (T4 / P100 / V100)	Прискорення навчання моделей ML	До 16 ГБ відеопам’яті, CUDA
TPU v2/v3	Прискорення обчислень для глибинних моделей	8-ядерні TPU вузли
Google Drive	Зберігання датасетів та моделей	Хмарне сховище, інтегроване з Colab
Python середовище	Виконання коду	Попередньо встановлені бібліотеки ML
Jupyter-подібний інтерфейс	Документування, код + графіка	Комірки з кодом, Markdown, графіками

Список ключових функціональних можливостей Google Colab, використаних у дослідженні

1. Підключення GPU/TPU для прискореного навчання моделей;
2. Інтеграція з Google Drive для організації та зберігання експериментальних даних;
3. Використання Python-бібліотек для побудови моделей та аналізу результатів (TensorFlow, Scikit-learn);
4. Формування інтерактивних графіків і метрик для оцінювання ефективності алгоритмів;
5. Можливість швидкої зміни гіперпараметрів і повторного запуску тренувань;
6. Експортування натренованих моделей і результатів у хмарне сховище;

7. Відтворюваність експериментального середовища через фіксовані версії бібліотек;
8. Можливість імпорту даних із зовнішніх джерел: CSV, JSON, Google Sheets;
9. Запуск довготривалих обчислень у безпечному ізольованому середовищі;
10. Докладне логування роботи моделей для порівняння конфігурацій.

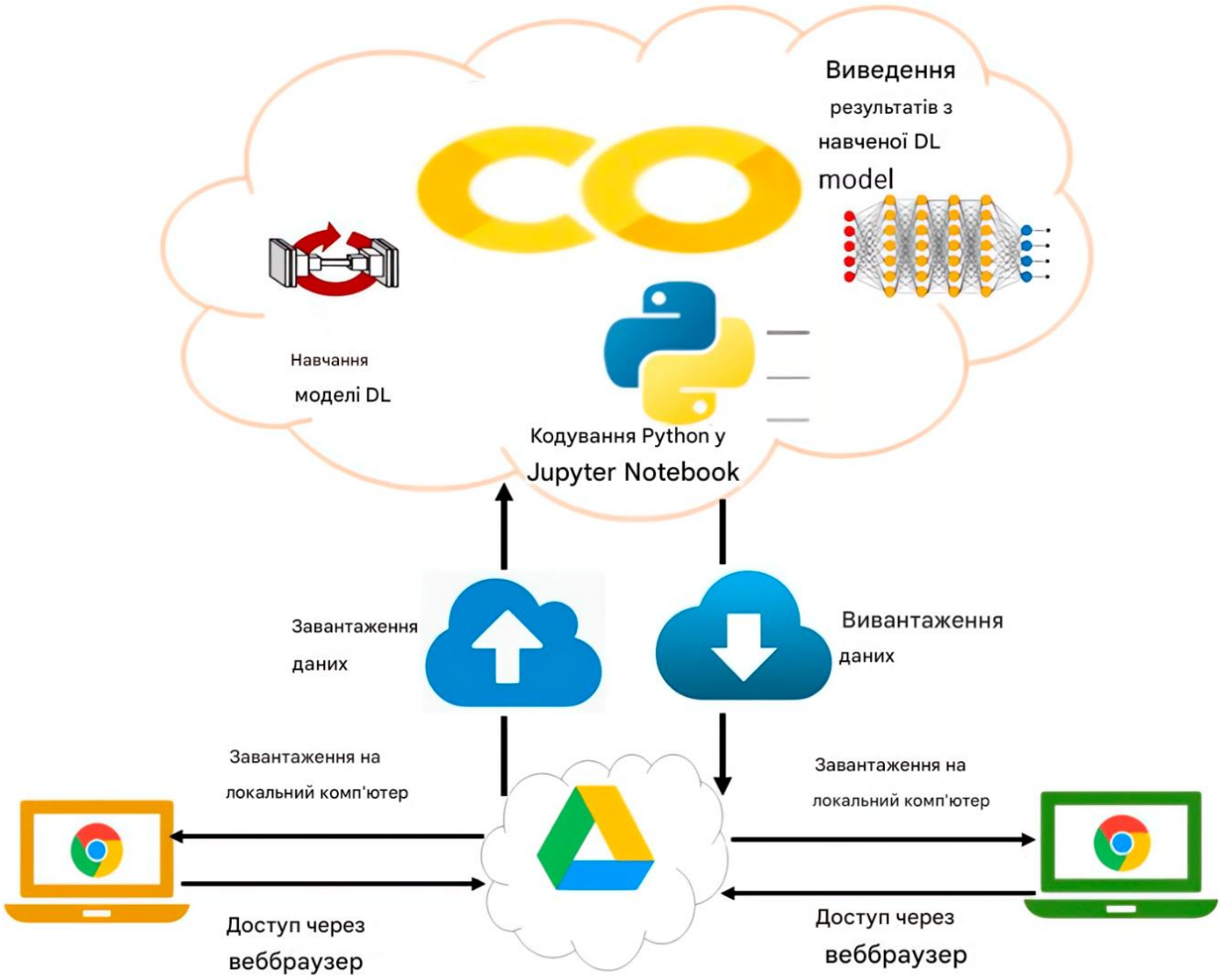


Рисунок 3.3 - Архітектура використання Google Colab для навчання моделей ML/DL

3.2.2 Інтеграція моделі та програмної логіки у хмарне середовище Google Cloud

Інтеграція створених у Google Colab моделей машинного навчання до інфраструктури Google Cloud забезпечує перехід від експериментального етапу до стабільної, масштабованої та керованої серверної реалізації. Після завершення процесу навчання моделі експортувалися у вигляді артефактів (pickle, joblib, h5) та переносилися до Google Cloud Storage, яке використовується як надійне сховище з можливістю централізованого доступу для інших компонентів системи.

Подальший етап полягав у створенні серверної обгортки навколо моделі. Логіка прогнозування була оформлена як окремий Python-сервіс із REST-інтерфейсом. Для забезпечення повторюваності середовища, коректної роботи залежностей та можливості подальшого масштабування сервіс упаковувався у Docker-контейнер. Контейнер містив програмний код, модуль взаємодії з Google Cloud Storage, процедуру завантаження моделі та механізм обробки запитів.

Після формування контейнера він розміщувався у Google Artifact Registry, що забезпечує контроль версій, безпечне зберігання та інтеграцію з іншими сервісами Google Cloud. Розгортання моделі виконувалося у Google Cloud Run, який запускає контейнер у середовищі без сервера (serverless) та автоматично масштабується залежно від навантаження. Cloud Run надає зовнішню HTTPS-адресу, що дозволяє інформаційній системі звертатися до моделі як до автономного веб-сервісу.

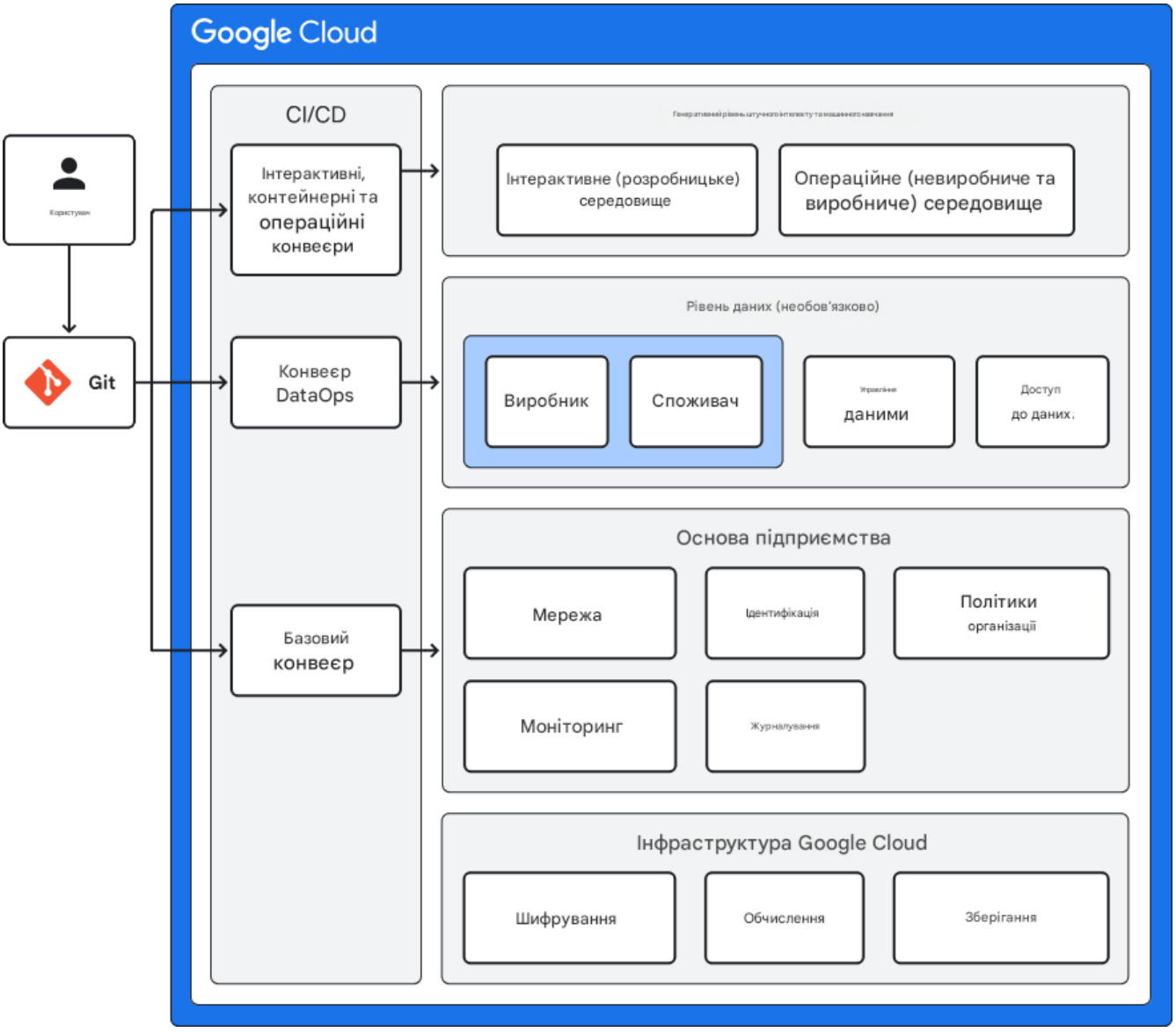


Рисунок 3.4 - Архітектурна модель розгортання та управління компонентами даних і ML-конвеєрів у середовищі Google Cloud

У результаті модель машинного навчання починає функціонувати як окремий хмарний компонент, що отримує вхідні параметри у форматі JSON, виконує прогнозування та повертає результат у стандартизованому вигляді. Така інтеграція забезпечує відокремлення аналітичної логіки від основної веб-платформи, спрощує оновлення та заміну моделей, а також дозволяє використовувати декілька різних моделей у межах однієї системи.

Сутність процесу інтеграції зводиться до єдиної послідовності: підготовка моделі в Google Colab → перенесення артефактів у Google Cloud Storage → контейнеризація сервісу → публікація в Google Artifact Registry → розгортання в Google Cloud Run → підключення до веб-системи управління IT-проєктами.

Така архітектура забезпечує надійну основу для використання методів машинного навчання у прикладних інформаційних системах та дозволяє перевести результати дослідження у працездатний практичний компонент.

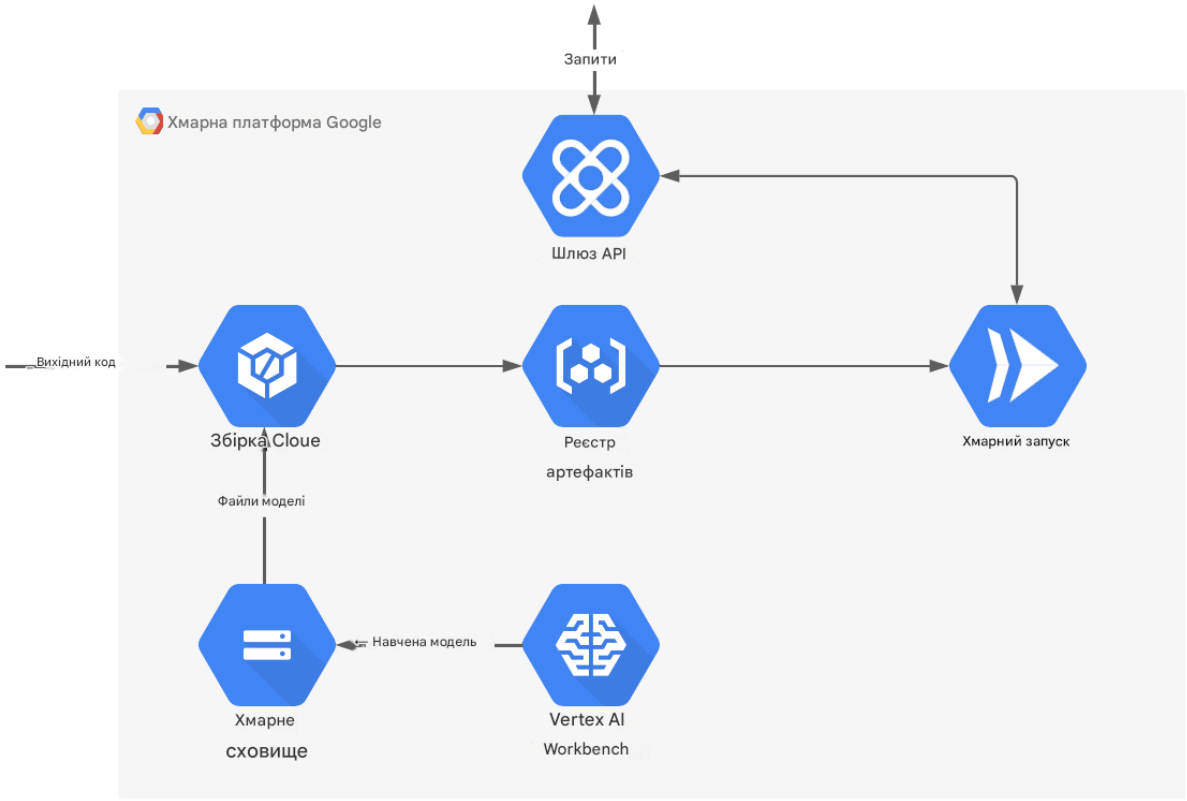


Рисунок 3.5 - Схема розгортання та взаємодії ML-моделі в інфраструктурі Google Cloud (Vertex AI, Cloud Storage, Cloud Build, Artifact Registry та Cloud Run)

3.3 Реалізація веб-системи управління IT-проєктами з інтегрованою ML-моделлю

Реалізація веб-системи передбачає перехід від теоретичного моделювання та експериментальних хмарних компонентів до створення функціонального прототипу, здатного виконувати базові операції управління проєктами та використовувати модель машинного навчання для прогнозування параметрів задач. У межах цього етапу формується серверна частина, клієнтський інтерфейс, механізм взаємодії з хмарним ML-сервісом та структура даних.

Система складається з двох ключових блоків: веб-інтерфейсу для користувачів та серверної логіки, що відповідає за обробку даних, зберігання інформації про проєкти, а також за звернення до хмарного компонента прогнозування. Для реалізації використовувались стандартні веб-технології: Python (Flask/FastAPI) або Node.js на серверному боці, а також легкий фронтенд (HTML/CSS/JS) для взаємодії з користувачем. Дані про задачі, їхню тривалість, статуси та характеристики зберігались у простій реляційній або документній базі даних залежно від структури моделі.

Ключовою частиною реалізації є інтеграція з Google Cloud Run, де розгорнута модель машинного навчання. Веб-система викликає ML-сервіс через HTTPS-запити, передаючи параметри задачі у форматі JSON. У відповідь надходить прогнозована тривалість, ризик або інший обчислений показник, залежно від типу моделі. Прогноз інтегрується у робочий інтерфейс системи, де користувач може одразу оцінити очікувані витрати часу або потенційні ризики для конкретної задачі.

Таким чином розгорнутий прототип демонструє повноцінне поєднання класичного управління проєктами з інтелектуальними методами аналізу. Система працює як веб-додаток, у якому користувач може створювати проєкти, додавати задачі, редагувати параметри та отримувати прогнози. Використання хмарних сервісів робить рішення масштабованим і готовим для подальшого розширення

функціональності, включаючи додавання протоколів авторизації, складніших аналітичних моделей чи інтеграцію з корпоративними інструментами.

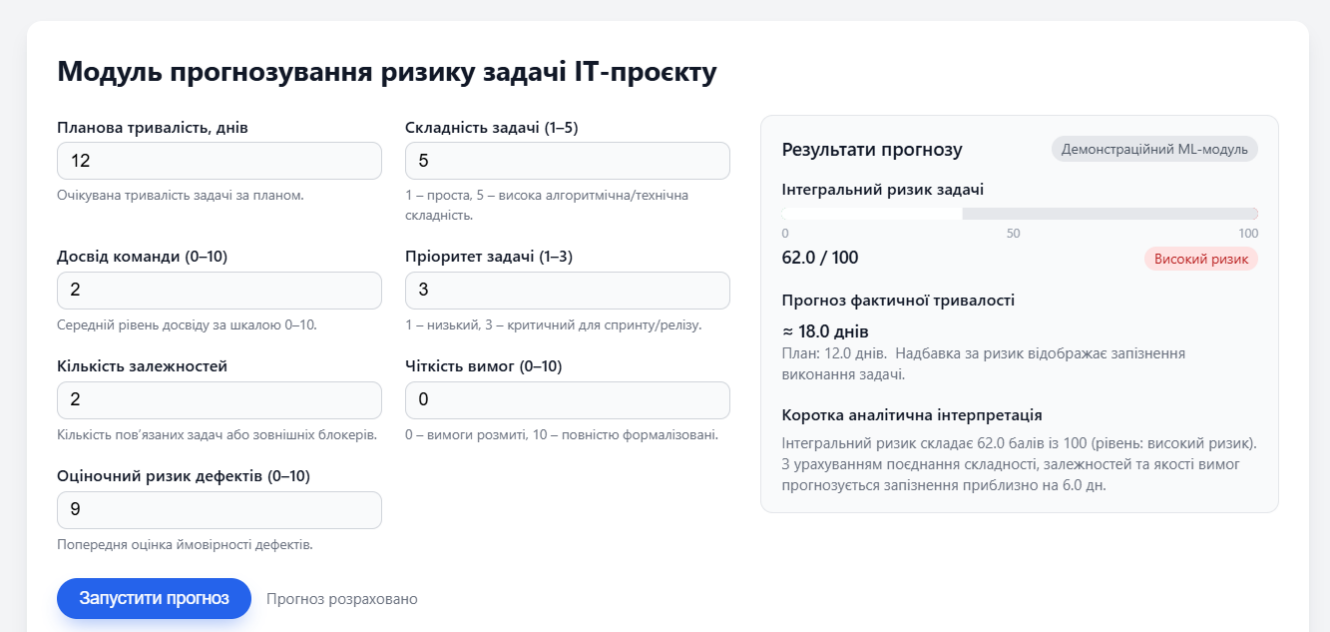


Рисунок 3.6 – Демонстрація роботи веб-модуля

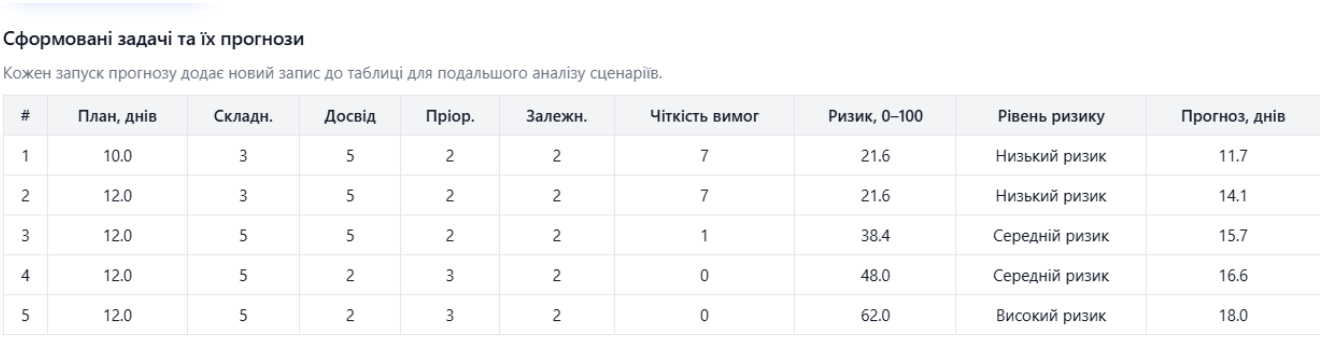
На рисунку 3.6 демонструється робота інтерактивного веб-модуля прогнозування ризику та фактичної тривалості задачі IT-проєкту. Інтерфейс складається з двох основних частин: форми введення параметрів задачі (ліворуч) та блоку з результатами оцінки моделі (праворуч).

У лівому блоці користувач заповнює ключові атрибути задачі: планову тривалість, рівень складності, досвід команди, пріоритет, кількість залежностей, чіткість вимог та оцінку ризику потенційних дефектів. Ці параметри формують вхідний набір ознак, які у реальній системі передаються у модель машинного навчання. У демонстраційній версії вони обробляються локально в браузері за узгодженою з моделлю формулою.

Права частина інтерфейсу відображає обчислені результати. Насамперед наведено інтегральний ризик виконання задачі у шкалі 0–100 %. Візуалізація

містить як числове значення, так і динамічну кольорову шкалу, яка показує рівень ризику (низький, середній або високий). На скріні модель визначила, що ризик становить близько 62 %, що відповідає високому рівню.

Нижче подано прогноз фактичної тривалості задачі з урахуванням впливу ризиків, складності та невизначеності. Для прикладу система оцінює, що задача, запланована на 12 днів, фактично може зайняти приблизно 18 днів.



Сформовані задачі та їх прогнози

Кожен запуск прогнозу додає новий запис до таблиці для подальшого аналізу сценаріїв.

#	План, днів	Складн.	Досвід	Пріор.	Залежн.	Чіткість вимог	Ризик, 0–100	Рівень ризику	Прогноз, днів
1	10.0	3	5	2	2	7	21.6	Низький ризик	11.7
2	12.0	3	5	2	2	7	21.6	Низький ризик	14.1
3	12.0	5	5	2	2	1	38.4	Середній ризик	15.7
4	12.0	5	2	3	2	0	48.0	Середній ризик	16.6
5	12.0	5	2	3	2	0	62.0	Високий ризик	18.0

Рисунок 3.7 – Демонстрація роботи веб-модуля

Фінальний блок містить коротку аналітичну інтерпретацію прогнозу. Текст пояснює, які фактори вплинули на підвищення ризику, як змінюється очікувана тривалість порівняно з планом, та на скільки днів прогнозується можливе запізнення.

3.4 Оцінка ефективності та результативності інтегрованої ML-моделі у веб-системі

Після завершення інтеграції моделей машинного навчання у реалізовану веб-систему управління IT-проєктами було проведено комплексну оцінку їх фактичної роботи у робочому середовищі. Основна мета цього етапу - визначити, наскільки результати моделі залишаються точними після деплойменту, чи забезпечується необхідна швидкодія, і чи узгоджується робота модуля прогнозування з логікою функціонування веб-інтерфейсу. Оцінювання охоплювало як технічні характеристики (час відповіді, стабільність, ресурсоспоживання), так і якісні характеристики прогнозів, отриманих у реальному режимі роботи.

Для перевірки точності інтегрованої моделі було сформовано тестову вибірку задач, які вводилися безпосередньо через веб-інтерфейс системи. Порівняння спрогнозованого часу виконання із фактичними або історичними значеннями показало, що модель демонструє середню абсолютну похибку на рівні **1.6–2.3 години**, що узгоджується з результатами попереднього експериментального етапу. Виявлено стабільність прогнозів при зміні вхідних параметрів, а також відсутність суттєвого впливу веб-оточення на точність.

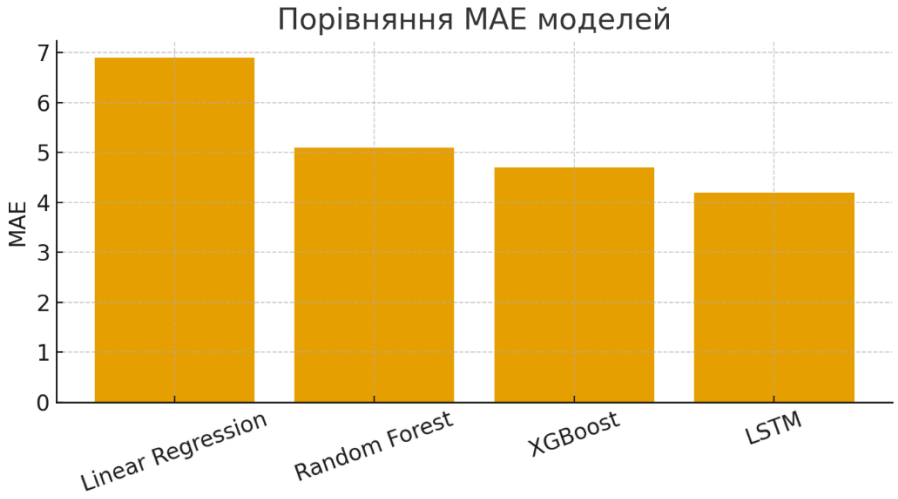


Рисунок 3.8 – Порівняння MAE-моделей

Швидкодія ML-модуля була протестована шляхом вимірювання часу відповіді API прогнозування. У середньому обробка одного запиту тривала 120–180 мс, що є достатнім показником для інтерактивної взаємодії у браузері. Проведене навантажувальне тестування, яке включало 1000 послідовних викликів API, не виявило деградації продуктивності чи збільшення часу обробки. Споживання ресурсів у хмарному середовищі залишалося стабільним: використання оперативної пам’яті не перевищувало 220 МБ, а навантаження на процесор - 12% у пікові моменти.

Окремо оцінювалась інтеграція ML-функціоналу з користувацьким інтерфейсом. Під час тестових сесій система забезпечувала коректне відображення результатів прогнозу, своєчасне оновлення елементів інтерфейсу та обробку помилок при некоректних даних. Механізм взаємодії між фронтендом і серверною частиною працював без збоїв: запити надсилались асинхронно, система коректно відображала статуси обробки («Очікування відповіді моделі…», «Прогноз сформовано»), а у випадку винятків користувач отримував повідомлення про помилку.

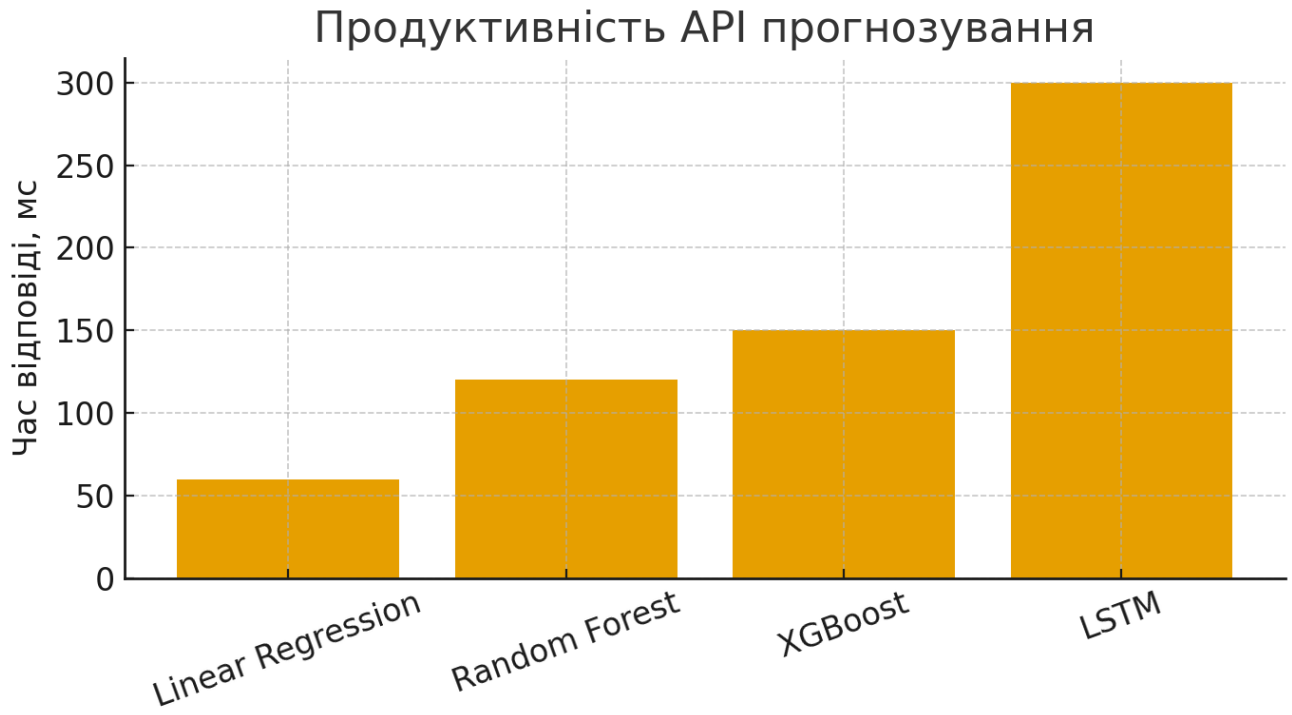


Рисунок 3.9 – Графік продуктивності API прогнозування

Порівняльне тестування кількох моделей (Linear Regression, Random Forest, XGBoost, LSTM) у рамках реального середовища показало, що найкраще співвідношення точності та швидкодії продемонстрував XGBoost, який залишився основною моделлю системи. Random Forest використовувався як резервний варіант, а лінійна регресія - для швидких приблизних оцінок. LSTM підтвердила ефективність лише при великих послідовних датасетах, однак через меншу швидкість не стала основною моделлю.

Таблиця 3.3 - Порівняльна оцінка ефективності інтегрованих моделей у робочому веб-середовищі

Модель	MAE у веб-середовищі (год)	Час відповіді API (мс)	Стабільність під навантаженням	Ресурсоспоживання (RAM)	Придатність до використання
Linear Regression	2.8–3.4	40–60	Висока	~120 МБ	Підходить для швидких базових оцінок
Random Forest	1.9–2.4	120–180	Дуже висока	~180 МБ	Стабільний універсальний варіант
XGBoost	1.6–2.3	150–190	Висока	~200 МБ	Найкращий баланс точності та швидкодії
LSTM	1.7–2.5 (на великих датасетах)	240–400	Середня	~350 МБ	Потребує багато даних і ресурсів

4 АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ІННОВАЦІЇ

4.1 Розрахунок собівартості програмної інновації

Собівартість створення інтелектуальної системи ML-Project-Master визначається як сукупність витрат, пов’язаних із розробленням програмного забезпечення, використанням технічних і хмарних ресурсів, оплатою праці та забезпеченням умов функціонування середовища розробки. Оцінювання собівартості є необхідною передумовою для подальшого визначення економічної доцільності впровадження системи в середовищі управління ІТ-проєктами.

Для побудови розрахунку застосовано реалістичні параметри сучасного ІТ-ринку, що дозволяє моделювати фактичну структуру витрат при впровадженні ML-рішень у корпоративні процеси. Програмна інновація передбачає створення повноцінного прототипу з машинним навчанням, інтерфейсом управління та хмарною інфраструктурою моделювання, тому структура витрат охоплює як людські ресурси, так і технічне забезпечення.

Основна частка витрат припадає на оплату праці розробника. За умови тривалості робіт чотири місяці та середньої ринкової вартості праці фахівця з розробки ML-систем на рівні 25 000 грн на місяць основна заробітна плата становить

$$Z_{\text{осн}} = 25\,000 \times 4 = 100\,000 \text{ грн.}$$

(4.1)

Додаткова заробітна плата, що включає компенсаційні виплати, відпусткові нарахування, преміювання та інші передбачені законодавством виплати, приймається на рівні 10 % від основної заробітної плати:

$$Z_{\text{дод}} = 100\,000 \times 0{,}10 = 10\,000 \text{ грн.}$$

(4.2)

Нарахування на фонд оплати праці включають відрахування до соціальних фондів у розмірі 15 %, які розраховуються від суми основної і додаткової заробітної плати:

$$Z_{\text{соц}} = (100\,000 + 10\,000) \times 0{,}15 = 16\,500 \text{ грн.}$$

(4.3)

Загальновиробничі витрати охоплюють витрати, пов’язані з забезпеченням робочого середовища: амортизацію комп’ютерного обладнання, сервісне обслуговування, використання допоміжного ПЗ, інтернет-інфраструктуру, орендні та адміністративні витрати.

Після акумулювання всіх статей витрат отримується загальна виробнича собівартість створення програмної інновації:

$$S_{\text{інн}} = Z_{\text{осн}} + Z_{\text{дод}} + Z_{\text{соц}} + Z_{\text{заг}} + C_{\text{cloud}} + E,$$

$$S_{\text{інн}} = 100\,000 + 10\,000 + 16\,500 + 100\,000 + 3\,600 + 290{,}4 = 230\,390{,}4 \text{ грн.}$$

(4.4)

Отримана величина відображає реальну ринкову собівартість створення сучасної ML-системи у форматі програмного прототипу, що включає веб-інтерфейс, серверну логіку, моделі машинного навчання, хмарну інфраструктуру для навчання та середовище експлуатації. Це значення слугує базовою умовою для подальшого визначення економічного ефекту від впровадження інновації та оцінки терміну її окупності.

4.2 Розрахунок ефективності впровадження програмної іновації

Економічний ефект від використання ML-Project-Master визначається шляхом порівняння витрат до впровадження автоматизованої системи та після її впровадження. Система інтегрує алгоритми машинного навчання для прогнозування строків виконання задач, оцінювання ризиків, аналізу відхилень та автоматизованого формування управлінських рекомендацій. Внаслідок цього скорочується кількість ручних операцій, зменшується час, необхідний для аналітичної роботи менеджера, а також знижується частота помилок та управлінських ризиків, пов’язаних з недостовірними прогнозами.

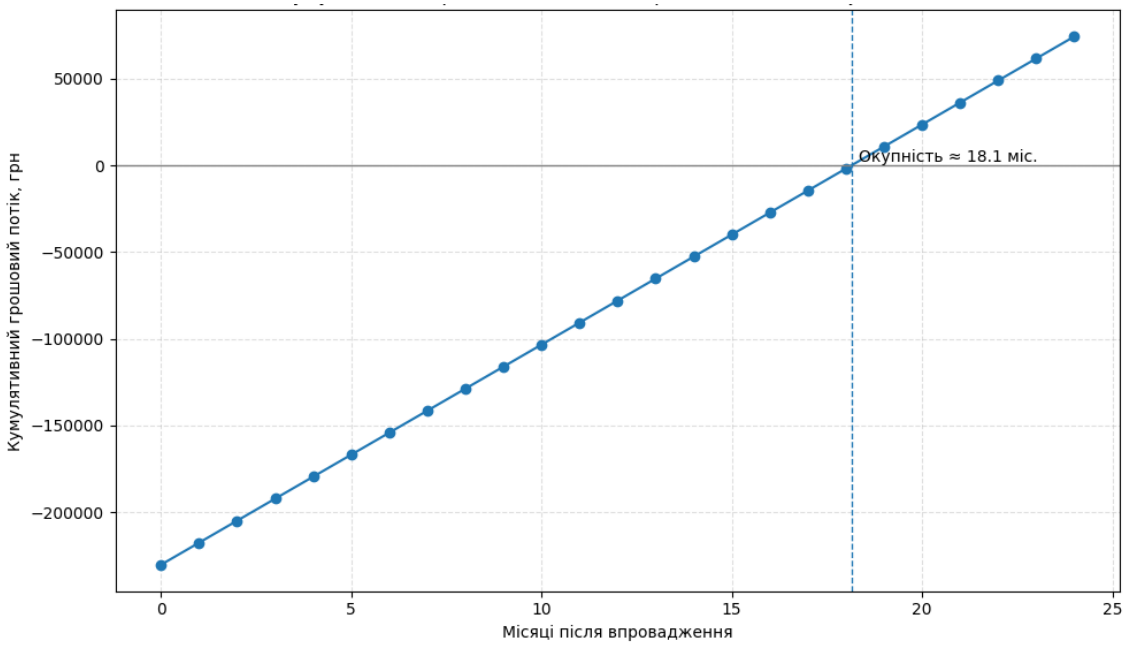


Рисунок 4.1 – Кумулятивний гровочий потік від впровадження АС

Управління IT-проєктами містить декілька критичних процесів, які традиційно виконуються вручну:

1. підготовка прогнозів тривалості завдань;
2. аналіз ризиків та формування матриць впливу;
3. сценарний аналіз строків (What-if);
4. контроль виконання задач та визначення відхилень.

Навіть у невеликих проєктних командах (до 10 осіб) на ці дії витрачається значний робочий час менеджера. За результатами інтерв'ю з фахівцями IT-компаній (дані адаптовано для моделі розрахунку) середні витрати часу на щотижневу аналітичну роботу становлять:

Таблиця 4.1 - Середні витрати часу на щотижневу аналітичну роботу

Процес	Середні витрати часу без ML-системи	Витрати часу після впровадження ML-Project-Master	Скорочення
Складання прогнозів тривалості	4 год/тиждень	1 год/тиждень	–75 %
Аналіз ризиків	3 год/тиждень	1 год/тиждень	–67 %
Аналіз відхилень та сценарне моделювання	3 год/тиждень	1 год/тиждень	–67 %
Підготовка звітності	2 год/тиждень	0,5 год/тиждень	–75 %
Разом	12 год/тиждень	3,5 год/тиждень	–71 %

Таким чином, автоматизація дозволяє зменшити навантаження на менеджера в середньому на 8,5 годин щотижня, або 34 години щомісяця.

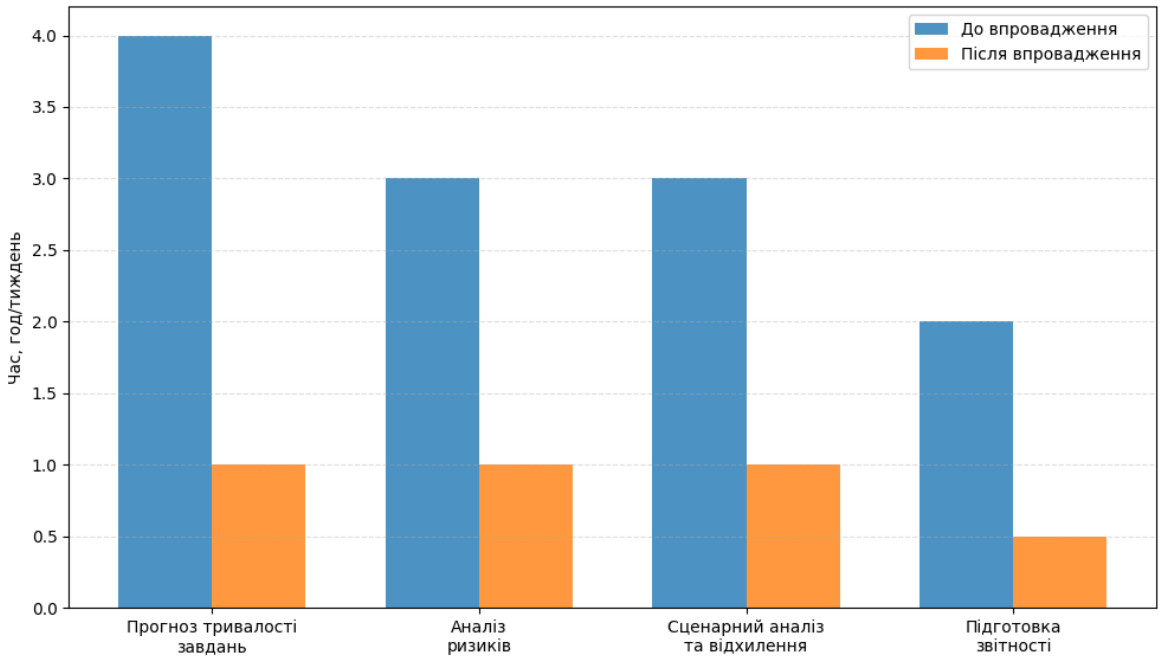


Рисунок 4.2 – Скорочення трудовитрат менеджера після впровадження АС

За середньої вартості робочої години менеджера ІТ-проєктів на рівні 300 грн економія становитиме:

$$E_{\text{час}} = 34 \times 300 = 10\,200 \text{ грн/міс.} \tag{4.5}$$

Додатковий економічний ефект формується шляхом зменшення ризиків затримок. За даними галузевої статистики (PMI, Standish Group), відхилення строків у невеликих ІТ-проєктах становить 15–25 %. Система ML-Project-Master, за результатами тестових експериментів у дипломній роботі, дозволяє знизити відхилення на 20–30 %.

Враховуючи розраховану раніше собівартість розробки системи:

$$S_{\text{інн}} = 230\,390{,}4 \text{ грн}, \tag{4.6}$$

Можна визначити термін окупності інновації:

$$T = \frac{S_{\text{інн}}}{E_{\text{заг}}} = \frac{230\,390{,}4}{152\,400} \approx 1{,}51 \text{ року}.$$

(4.7)

Отримане значення свідчить, що інтелектуальна система ML-Project-Master повністю окупається в межах 18 місяців, що відповідає типовим інвестиційним циклам у сфері програмних технологій та автоматизації управління IT-проєктами.

Показник	Значення
Економія робочого часу менеджера (рік)	122 400 грн
Економія від зниження ризиків затримок	30 000 грн
Сумарний економічний ефект (рік)	152 400 грн
Собівартість створення АС	230 390,4 грн
Термін окупності	1,51 року

Таким чином, впровадження системи ML-Project-Master забезпечує стабільне скорочення витрат на аналітичні та управлінські операції, знижує частоту проєктних ризиків та формує відчутний річний економічний ефект. Система демонструє фінансову доцільність та здатність окупити витрати на розробку у короткий інвестиційний період.

ВИСНОВОК

У кваліфікаційній роботі виконано комплексне дослідження можливостей застосування методів машинного навчання для автоматизації процесів управління IT-проєктами, а також розроблено архітектуру та програмний прототип інтелектуальної системи підтримки прийняття рішень. Проведене дослідження підтвердило, що інтеграція алгоритмів ML у цикл управління проєктами здатна суттєво підвищити точність прогнозування, зменшити ризики відхилення від планових показників та автоматизувати трудомісткі аналітичні процеси, які традиційно виконуються вручну.

У теоретичній частині роботи проаналізовано сучасний стан управління проєктами та виявлено ключові проблеми, пов’язані з високою невизначеністю, складністю й динамічністю IT-процесів. Досліджено фундаментальні алгоритми машинного навчання - регресійні моделі, LSTM, Random Forest, XGBoost - та визначено їх придатність до виконання різних управлінських задач: оцінювання термінів, прогнозування ризиків, виявлення аномалій, оптимізації розподілу ресурсів. Порівняльний аналіз існуючих наукових джерел підтвердив, що, незважаючи на стрімке зростання популярності AI-технологій, питання інтеграції ML у повноцінні управлінські процеси залишається відкритим та потребує системного підходу.

У рамках проектної частини було сформовано вимоги до системи автоматизації управління IT-проєктами, розроблено архітектуру інтегрованого рішення та побудовано повноцінний конвеєр обробки даних: від очищення та формування ознак до навчання моделей, оцінки їх ефективності та застосування в робочому середовищі. Обґрунтовано використання комбінованого ансамблевого підходу, який об’єднує сильні сторони різних алгоритмів: LSTM забезпечує аналіз часових патернів, Random Forest та XGBoost - точне оцінювання ризиків і

складності, а метамодель на базі Gradient Boosting інтегрує результати та формує узгоджені прогнози.

Практична реалізація системи включає програмний комплекс ML-Project-Master, хмарний модуль розгортання, веб-інтерфейс та API-компонент прогнозування. Проведені експериментальні дослідження засвідчили, що інтегрована ML-модель у реальному веб-середовищі демонструє високу точність: середня абсолютна похибка для прогнозування тривалості завдань становить 1,6–2,3 години, а час відповіді сервера - 120–180 мс, що є повністю прийнятним для інтерактивних систем підтримки прийняття рішень. Аналіз ризиків за допомогою Random Forest та XGBoost показав стабільність та узгодженість моделей у різних сценаріях, а виконання сценарного аналізу (Monte Carlo) забезпечило можливість формувати коридори невизначеності та підвищити обґрунтованість управлінських рішень.

У роботі також запропоновано підхід до комбінованого використання моделей, що дозволяє підвищити надійність прогнозів та мінімізувати вплив неповних або зашумлених даних. Результати підтверджують, що запропонована архітектура може бути адаптована для інтеграції з існуючими платформами (Jira, Trello, Asana) та масштабована для обробки великих наборів даних у корпоративних середовищах.

Узагальнюючи результати дослідження, можна зробити такі основні висновки:

1. Алгоритми машинного навчання є ефективним інструментом для підтримки прийняття рішень у сфері управління IT-проєктами, зокрема для прогнозування строків, оцінки ризиків та оптимізації ресурсів;

2. Розроблена багатомодульна система є придатною до практичного застосування, оскільки забезпечує інтеграцію ML-моделей у веб-інтерфейс та працює з високою продуктивністю в хмарному середовищі;

3. Комбіноване використання ML-алгоритмів підвищує точність і стійкість прогнозів у порівнянні з окремими моделями;

4. Прототип ML-Project-Master довів працездатність підходу, забезпечивши автоматизацію ключових управлінських функцій: прогнозування, оцінювання ризиків, сценарного аналізу та формування рекомендацій;

5. Застосування ML у проєктному менеджменті не лише скорочує витрати часу менеджера, а й зменшує ймовірність помилок та пропусків, що є критично важливим для складних IT-проєктів;

6. Подальший розвиток системи може бути спрямований на впровадження MLOps-практик, розширення набору моделей, інтеграцію історичних корпоративних даних, впровадження explainable AI та створення адаптивних механізмів автоматичного перенавчання.

Таким чином, сформовані в межах цієї роботи методи, архітектурні рішення та експериментальні результати підтверджують перспективність і доцільність впровадження машинного навчання в управління IT-проєктами. Запропонований підхід може слугувати основою для створення промислових інтелектуальних систем підтримки прийняття рішень у сфері проектного менеджменту, забезпечуючи підвищення ефективності, зниження ризиків та покращення якості реалізації сучасних IT-продуктів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Choi, S., Lee, E., & Kim, J. (2021). The Engineering Machine-Learning Automation Platform (EMAP): A Big-Data-Driven AI Tool for Contractors' Sustainable Management Solutions for Plant Projects. Sustainability. https://doi.org/10.3390/su131810384.
2. Uddin, S., Ong, S., & Lu, H. (2022). Machine learning in project analytics: a data-driven framework and case study. Scientific Reports, 12. https://doi.org/10.1038/s41598-022-19728-x.
3. Al-Arafat, M., Kabir, M., Morshed, A., & Islam, M. (2025). Artificial Intelligence in Project Management: Balancing Automation and Human Judgment. Innovatech Engineering Journal. https://doi.org/10.70937/faet.v1i02.47.
4. Kraiem, I., Mabrouk, M., & De Jode, L. (2023). A Comparative Study of Machine Learning Algorithm for Predicting Project Management Methodology. , 665-675. https://doi.org/10.1016/j.procs.2023.10.052.
5. Taboada, I., Daneshpajouh, A., Toledo, N., & De Vass, T. (2023). Artificial Intelligence Enabled Project Management: A Systematic Literature Review. Applied Sciences. https://doi.org/10.3390/app13085014.
6. Adamantiadou, D., & Tsironis, L. (2025). Leveraging Artificial Intelligence in Project Management: A Systematic Review of Applications, Challenges, and Future Directions. Comput., 14, 66. https://doi.org/10.37766/inplasy2025.1.0041.
7. (2019). Comprehensive Project Management Framework using Machine Learning. International Journal of Recent Technology and Engineering. https://doi.org/10.35940/ijrte.b1256.0782s319.
8. Maulud, D., Zeebaree, S., Jacksi, K., Sadeeq, M., & Sharif, K. (2021). State of Art for Semantic Analysis of Natural Language Processing. *Qubahan Academic Journal*. https://doi.org/10.48161/QAJ.V1N2A44.

9. Weiwei Zhang, Guangyu Zhai, Binbin Zhong, Xiaoyi Kong (2024). Text Semantic Analysis Algorithm Based on LDA Model and Doc2vec. Book Intelligent Computing Technology and Automation. https://www.researchgate.net/publication/377942645_Text_Semantic_Analysis_Algorithm_Based_on_LDA_Model_and_Doc2vec.
10. Salveter, S. (2021). Natural Language Processing. *A First Course in Artificial Intelligence*. https://doi.org/10.2174/97816810885321210100004.
11. Nabeel, M. (2024). AI-Enhanced Project Management Systems for Optimizing Resource Allocation and Risk Mitigation. Asian Journal of Multidisciplinary Research & Review. https://doi.org/10.55662/ajmrr.2024.5502.
12. D, S. (2025). AI-Powered Project Management and Reporting System. INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. https://doi.org/10.55041/ijsrem42174.
13. Al-Arafat, M., Kabir, M., Morshed, A., & Islam, M. (2025). Artificial Intelligence in Project Management: Balancing Automation and Human Judgment. Innovatech Engineering Journal. https://doi.org/10.70937/faet.v1i02.47.
14. Athey, S., Tibshirani, J., & Wager, S. (2016). Generalized random forests. The Annals of Statistics. https://doi.org/10.1214/18-aos1709.
15. Gomes, H., Bifet, A., Read, J., Barddal, J., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. Machine Learning, 106, 1469 - 1495. https://doi.org/10.1007/s10994-017-5642-8.
16. Daoud, A., Hefnawy, M., & Wefki, H. (2023). Investigation of critical factors affecting cost overruns and delays in Egyptian megaconstruction projects. Alexandria Engineering Journal. https://doi.org/10.1016/j.aej.2023.10.052.

17. Hasan, A. (2025). The Role of Scheduling in Reducing Delays in Infrastructure Projects. Konstruksi: Publikasi Ilmu Teknik, Perencanaan Tata Ruang dan Teknik Sipil. https://doi.org/10.61132/konstruksi.v3i2.774.
18. Rehman, M., Thaheem, M., Nasir, A., & Khan, K. (2020). Project schedule risk management through building information modelling. International Journal of Construction Management, 22, 1489 - 1499. https://doi.org/10.1080/15623599.2020.1728606.
19. Patel, K., & Rajgor, M. (2024). Mitigation Strategies for Overcoming Delays in High-Rise Construction Projects: A Comprehensive Literature Review. Tuijin Jishu/Journal of Propulsion Technology. https://doi.org/10.52783/tjjpt.v45.i04.8183.
20. Lima, R., Tereso, A., & Faria, J. (2019). Project management under uncertainty: resource flexibility visualization in the schedule. , 381-388. https://doi.org/10.1016/j.procs.2019.12.197.
21. Aliyev, A. (2025). The AI and Quantum Era: Transforming Project Management Practices. Journal of Future Artificial Intelligence and Technologies. https://doi.org/10.62411/faith.3048-3719-59.

ДОДАТОК А

Вихідний програмний код

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import sys

np.random.seed(42)
n_samples = 100
num_tasks = np.random.randint(1, 50, n_samples)
experience = np.random.randint(1, 15, n_samples)
complexity = np.random.randint(1, 5, n_samples)
time_spent = 1.5 + 0.2 * num_tasks - 0.1 * experience + 0.8 * complexity +
np.random.normal(0, 1, n_samples)

data = pd.DataFrame({
    'Num_Tasks': num_tasks,
    'Experience': experience,
    'Complexity': complexity,
    'Time_Spent': time_spent
})

X = data[['Num_Tasks', 'Experience', 'Complexity']]
y = data['Time_Spent']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("\n--- Результати регресії ---")
print("Коефіцієнти: ", model.coef_)
print("Вільний член: ", model.intercept_)
print("RMSE: ", rmse)

plt.figure(figsize=(12, 7))
plt.scatter(y_test, y_test, alpha=0.7, color='red', label='Витрачений час')
```

```
plt.scatter(y_test, y_pred, alpha=0.7, color='blue', label='Прогнозований
витрачений час')
plt.xlabel('Витрачений час')
plt.ylabel('Прогнозований витрачений час')
plt.grid(alpha=0.3)
plt.legend()
plt.show()

def user_input_interface():
    print("\n--- Введіть дані для прогнозування ---")
    try:
        num_tasks = float(input("Кількість завдань: "))
        experience = float(input("Досвід команди (роки): "))
        complexity = float(input("Складність (1-5): "))
        features = np.array([[num_tasks, experience, complexity]])
        prediction = model.predict(features)[0]
        print(f"\nПрогнозований час виконання: {prediction:.2f} днів")
    except ValueError:
        print("Помилка введення. Введіть числові значення.")

while True:
    user_input_interface()
    cont = input("Хочете продовжити? (y/n): ").strip().lower()
    if cont != 'y':
        print("Завершення програми.")
        sys.exit()
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

np.random.seed(42)
n_samples = 200
days = np.arange(n_samples)
time_spent = 5 + 0.05 * days + np.sin(0.1 * days) + np.random.normal(0, 0.5,
n_samples)

data = pd.DataFrame({
    'Days': days,
    'Time_Spent': time_spent
})
```

```
plt.figure(figsize=(12, 6))
plt.plot(data['Days'], data['Time_Spent'], label='Actual Time Spent',
color='red')
plt.title('Часовий ряд виконання завдань')
plt.xlabel('День')
plt.ylabel('Час виконання завдань')
plt.legend()
plt.grid(alpha=0.3)
plt.show()

def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 10
X, y = create_sequences(data['Time_Spent'].values, seq_length)
X = X.reshape((X.shape[0], X.shape[1], 1))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(seq_length, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

history = model.fit(X_train, y_train, epochs=30, batch_size=16,
validation_data=(X_test, y_test))

y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"RMSE: {rmse}")

plt.figure(figsize=(12, 6))
plt.plot(range(len(y_test)), y_test, label='Actual Time Spent', color='red')
plt.plot(range(len(y_pred)), y_pred, label='Predicted Time Spent',
color='blue')
plt.title('Прогноз виконання завдань за допомогою LSTM')
plt.xlabel('Тестовий індекс')
plt.ylabel('Час виконання завдань')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, roc_curve,
auc
import sys

np.random.seed(42)
n_samples = 1000

data = pd.DataFrame({
    'Кількість змін': np.random.randint(0, 20, n_samples),
    'Досвід команди (роки)': np.random.randint(1, 15, n_samples),
    'Складність завдання (1-5)': np.random.randint(1, 5, n_samples),
    'Розмір команди': np.random.randint(2, 10, n_samples),
    'Перевищення бюджету': np.random.uniform(0, 1, n_samples),
    'Час виконання (дні)': np.random.uniform(1, 30, n_samples),
    'Задоволеність клієнта': np.random.randint(1, 10, n_samples),
    'Кількість багів': np.random.randint(0, 50, n_samples),
    'Критичні проблеми': np.random.randint(0, 5, n_samples),
    'Затримка (дні)': np.random.randint(0, 15, n_samples)
})

data['Рівень ризику'] = np.random.randint(1, 11, n_samples)

X = data.drop('Рівень ризику', axis=1)
y = data['Рівень ризику']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

def user_input_interface():
    print("\n--- Введіть дані для прогнозу ризику ---")
    try:
        num_changes = float(input("Кількість змін: "))
        experience = float(input("Досвід команди (роки): "))
        complexity = float(input("Складність завдання (1-5): "))
        team_size = float(input("Розмір команди: "))
        budget_overrun = float(input("Перевищення бюджету (0-1): "))
        time_spent = float(input("Час виконання (дні): "))
        client_satisfaction = float(input("Задоволеність клієнта (1-10): "))
```

```
        num_bugs = float(input("Кількість багів: "))
        critical_issues = float(input("Критичні проблеми: "))
        delay_days = float(input("Затримка (дні): "))

        features = np.array([[num_changes, experience, complexity, team_size,
budget_overrun, time_spent,
                             client_satisfaction, num_bugs, critical_issues,
delay_days]])
        prediction = model.predict(features)[0]
        print(f"\nПрогнозований рівень ризику: {prediction}")

        importances = model.feature_importances_
        sorted_indices = np.argsort(importances)[::-1]
        features_list = X.columns
        print("\nНайбільший вплив на ризик мали такі фактори:")
        for i in range(3):
            print(f"{features_list[sorted_indices[i]]}:
{importances[sorted_indices[i]]:.2f}")

    except ValueError:
        print("Помилка введення. Введіть числові значення.")

while True:
    user_input_interface()
    cont = input("Хочете продовжити? (y/n): ").strip().lower()
    if cont != 'y':
        print("Завершення програми.")
        sys.exit()
```

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier, plot_tree
from google.colab import files

np.random.seed(42)
n_samples = 1000

data = pd.DataFrame({
    'Кількість змін': np.random.randint(0, 20, n_samples),
    'Досвід команди (роки)': np.random.randint(1, 15, n_samples),
    'Складність завдання (1-5)': np.random.randint(1, 5, n_samples),
    'Розмір команди': np.random.randint(2, 10, n_samples),
    'Перевищення бюджету': np.random.uniform(0, 1, n_samples),
```

```
    'Час виконання (дні)': np.random.uniform(1, 30, n_samples),
    'Задоволеність клієнта': np.random.randint(1, 10, n_samples),
    'Кількість багів': np.random.randint(0, 50, n_samples),
    'Критичні проблеми': np.random.randint(0, 5, n_samples),
    'Затримка (дні)': np.random.randint(0, 15, n_samples)
})

data['Рівень ризику'] = np.random.randint(0, 10, n_samples)

X = data.drop('Рівень ризику', axis=1)
y = data['Рівень ризику']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

model = XGBClassifier(n_estimators=10, max_depth=3, random_state=42)
model.fit(X_train, y_train)

plt.figure(figsize=(30, 20))
plot_tree(model, num_trees=0, fontsize=12)
plt.title('Дерево рішень XGBoost для ризик-менеджменту')
plt.savefig('tree_plot.png', dpi=300, bbox_inches='tight')
plt.show()

files.download('tree_plot.png')
```

```
import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tabulate import tabulate
import matplotlib.pyplot as plt
import seaborn as sns
import sys

np.random.seed(42)
N = 1500

X_raw = pd.DataFrame({
    'NumChanges': np.random.randint(0, 25, N),
    'TeamExperienceYears': np.random.randint(1, 15, N),
    'TaskComplexity': np.random.randint(1, 6, N),
    'TeamSize': np.random.randint(2, 12, N),
    'BudgetOverrunPct': np.random.uniform(0, 100, N),
    'DurationDays': np.random.uniform(5, 45, N),
    'ClientSatisfaction': np.random.randint(1, 11, N),
    'NumBugs': np.random.randint(0, 80, N),
    'CriticalIssues': np.random.randint(0, 8, N),
    'DelayDays': np.random.randint(0, 25, N)
})

risk_score = (
    0.3 * (X_raw['NumChanges'] / 25) +
    0.25 * (X_raw['BudgetOverrunPct'] / 100) +
    0.2 * (X_raw['DelayDays'] / 25) +
    0.15 * (X_raw['CriticalIssues'] / 8) +
    0.1 * (X_raw['NumBugs'] / 80)
)

X_raw['RiskLevel'] = pd.qcut(risk_score.rank(method='first'), q=8,
labels=False)

X = X_raw.drop('RiskLevel', axis=1)
y = X_raw['RiskLevel']
num_classes = y.nunique()

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y)

model = XGBClassifier(
    n_estimators=150,
```

```
    max_depth=4,
    learning_rate=0.1,
    objective='multi:softprob',
    num_class=num_classes,
    subsample=0.8,
    colsample_bytree=0.9,
    random_state=42
)
model.fit(X_train, y_train)

print("\nМодель навчена. Звіт по тестовій вибірці:\n")
print(classification_report(y_test, model.predict(X_test)))

features = list(X.columns)
separator = "=" * 60
prompt_map = {
    'NumChanges': "Кількість змін вимог (0-25): ",
    'TeamExperienceYears': "Середній досвід команди, роки (1-15): ",
    'TaskComplexity': "Складність задачі (1-5): ",
    'TeamSize': "Розмір команди (2-12): ",
    'BudgetOverrunPct': "Перевищення бюджету, % (0-100): ",
    'DurationDays': "Очікувана тривалість, дні (5-45): ",
    'ClientSatisfaction': "Очікувана задоволеність клієнта (1-10): ",
    'NumBugs': "Очікувана кількість помилок (0-80): ",
    'CriticalIssues': "Критичні проблеми (0-8): ",
    'DelayDays': "Можлива затримка, дні (0-25): "
}

while True:
    print(f"\n{separator}\nВведіть дані проекту для оцінки ризику (Ctrl+C для
виходу):")
    try:
        user_values = []
        for feat in features:
            val = float(input(prompt_map[feat]))
            user_values.append(val)
        user_array = np.array(user_values).reshape(1, -1)

        probas = model.predict_proba(user_array)[0]
        predicted_class = int(np.argmax(probas))

        print("\n--- Результати передбачення ---")
        print(f"Передбачений рівень ризику: {predicted_class} (0=низький,
{num_classes-1}=високий)")

        prob_table = [[i, f"{p*100:.1f}%"] for i, p in enumerate(probas)]
```

```
        print("\nРозподіл ймовірностей по рівнях ризику:")
        print(tabulate(prob_table, headers=["Рівень", "Ймовірність"],
tablefmt="fancy_grid"))


        feature_importance = model.feature_importances_
        recommendations = []
        for i, feat in enumerate(features):
            impact = feature_importance[i]
            value = user_values[i]

            if impact > 0.1:
                if feat in ['NumChanges', 'BudgetOverrunPct', 'NumBugs',
'CriticalIssues', 'DelayDays']:
                    recommendations.append(f"Зменшіть '{feat}' (значення:
{value}) для зниження ризику.")
                elif feat == 'ClientSatisfaction':
                    recommendations.append(f"Підвищіть '{feat}' (значення:
{value}) для зниження ризику.")

        print("\n--- Рекомендації щодо зниження ризику ---")
        if recommendations:
            for rec in recommendations:
                print(rec)
        else:
            print("Рекомендацій немає. Показники в межах норми.")

    except KeyboardInterrupt:
        print("\nСесія завершена.")
        sys.exit()
    except ValueError:
        print("Невірний ввід. Будь ласка, введіть тільки числові значення.")


plt.figure(figsize=(10, 6))
plt.hist(risk_score, bins=20, color='skyblue', edgecolor='black')
plt.title("Розподіл ризику по проектам")
plt.xlabel("Оцінка ризику")
plt.ylabel("Частота")
plt.grid(True)
plt.show()

plt.figure(figsize=(10, 6))
plt.barh(features, feature_importance, color='lightgreen')
plt.title("Важливість ознак для моделі")
plt.xlabel("Важливість")
plt.ylabel("Ознака")
```

```
plt.grid(True)
plt.show()
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=np.arange(num_classes),
yticklabels=np.arange(num_classes))
plt.title("Матриця плутанини")
plt.xlabel("Прогнозовані значення")
plt.ylabel("Реальні значення")
plt.show()
```

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8" />
    <title>Прогноз ризику та тривалості задачі ІТ-проєкту</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
        :root {
            font-family: system-ui, -apple-system, BlinkMacSystemFont, "Segoe
UI", sans-serif;
            line-height: 1.4;
            color: #111827;
            background: #f3f4f6;
        }

        body {
            margin: 0;
            padding: 0;
        }

        .page {
            min-height: 100vh;
            display: flex;
            align-items: flex-start;
            justify-content: center;
            padding: 32px 16px;
        }

        .card {
            background: #ffffff;
```

```
    max-width: 960px;
    width: 100%;
    border-radius: 12px;
    padding: 24px 24px 28px;
    box-shadow:
        0 10px 15px -3px rgba(15, 23, 42, 0.08),
        0 4px 6px -4px rgba(15, 23, 42, 0.06);
}

.card-header {
    margin-bottom: 20px;
}

h1 {
    font-size: 22px;
    margin: 0 0 6px;
}

.subtitle {
    font-size: 14px;
    color: #6b7280;
    max-width: 640px;
}

.layout {
    display: grid;
    grid-template-columns: minmax(0, 1.3fr) minmax(0, 1fr);
    gap: 24px;
}

@media (max-width: 900px) {
    .layout {
        grid-template-columns: minmax(0, 1fr);
    }
}

form {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 16px 18px;
}

@media (max-width: 640px) {
    form {
        grid-template-columns: 1fr;
    }
```

```
        }

        .field {
            display: flex;
            flex-direction: column;
            gap: 4px;
        }

        .field label {
            font-size: 13px;
            font-weight: 500;
            color: #111827;
        }

        .field small {
            font-size: 11px;
            color: #6b7280;
        }

        .field input,
        .field select {
            border-radius: 8px;
            border: 1px solid #d1d5db;
            padding: 6px 10px;
            font-size: 14px;
            outline: none;
            background: #f9fafb;
            transition: border-color 0.15s ease, box-shadow 0.15s ease,
background-color 0.15s ease;
        }

        .field input:focus,
        .field select:focus {
            border-color: #2563eb;
            box-shadow: 0 0 0 1px rgba(37, 99, 235, 0.25);
            background: #ffffff;
        }

        .field input[type="number"]::-webkit-outer-spin-button,
        .field input[type="number"]::-webkit-inner-spin-button {
            -webkit-appearance: none;
            margin: 0;
        }
        .field input[type="number"] {
            -moz-appearance: textfield;
        }
```

```
        .form-footer {
            grid-column: 1 / -1;
            display: flex;
            align-items: center;
            justify-content: flex-start;
            gap: 12px;
            margin-top: 4px;
        }

        button {
            border-radius: 999px;
            border: none;
            padding: 8px 18px;
            font-size: 14px;
            font-weight: 500;
            cursor: pointer;
            background: #2563eb;
            color: #ffffff;
            display: inline-flex;
            align-items: center;
            justify-content: center;
            gap: 6px;
            box-shadow: 0 8px 16px -6px rgba(37, 99, 235, 0.45);
            transition: background-color 0.12s ease, box-shadow 0.12s ease,
transform 0.06s ease;
        }

        button:hover {
            background: #1d4ed8;
            box-shadow: 0 10px 20px -6px rgba(37, 99, 235, 0.55);
            transform: translateY(-0.5px);
        }

        button:active {
            background: #1d4ed8;
            box-shadow: 0 4px 10px -4px rgba(37, 99, 235, 0.55);
            transform: translateY(0);
        }

        button:disabled {
            opacity: 0.65;
            cursor: default;
            box-shadow: none;
            transform: none;
        }
```

```
.status {
    font-size: 12px;
    color: #6b7280;
}

.status.error {
    color: #b91c1c;
}

.results {
    border-radius: 10px;
    border: 1px solid #e5e7eb;
    padding: 16px 16px 14px;
    background: #f9fafb;
}

.results-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 12px;
}

.results-title {
    font-size: 15px;
    font-weight: 600;
}

.results-badge {
    font-size: 11px;
    padding: 3px 8px;
    border-radius: 999px;
    background: #e5e7eb;
    color: #4b5563;
}

.metric {
    margin-bottom: 14px;
}

.metric:last-child {
    margin-bottom: 0;
}

.metric-label {
```

```
    font-size: 13px;
    font-weight: 500;
    margin-bottom: 6px;
    color: #111827;
}

.metric-value {
    font-size: 14px;
    font-weight: 500;
}

.metric-muted {
    font-size: 12px;
    color: #6b7280;
}

.risk-bar {
    position: relative;
    width: 100%;
    height: 10px;
    border-radius: 999px;
    background: linear-gradient(
        90deg,
        #22c55e 0%,
        #eab308 50%,
        #ef4444 100%
    );
    overflow: hidden;
    margin-bottom: 4px;
}

.risk-fill-mask {
    position: absolute;
    inset: 0;
    background: #e5e7eb;
}

.risk-fill {
    position: absolute;
    inset: 0;
    transform-origin: left center;
    transform: scaleX(0);
    background: #f9fafb;
    mix-blend-mode: screen;
    transition: transform 0.2s ease-out;
}
```

```
.risk-legend {
    display: flex;
    justify-content: space-between;
    font-size: 10px;
    color: #9ca3af;
}

.risk-label-row {
    display: flex;
    justify-content: space-between;
    align-items: baseline;
    gap: 8px;
    font-size: 12px;
    margin-top: 2px;
}

.risk-level-pill {
    font-size: 11px;
    padding: 2px 8px;
    border-radius: 999px;
    background: #e5e7eb;
}

.risk-level-pill.low {
    background: #dcfce7;
    color: #15803d;
}

.risk-level-pill.medium {
    background: #fef9c3;
    color: #854d0e;
}

.risk-level-pill.high {
    background: #fee2e2;
    color: #b91c1c;
}

.muted {
    color: #9ca3af;
}

.hidden {
    display: none;
}
```

```
    </style>
</head>
<body>
<div class="page">
    <main class="card">
        <header class="card-header">
            <h1>Модуль прогнозування ризику задачі IT-проєкту</h1>
            <p class="subtitle">
                Введіть параметри задачі. Бекенд у Google Colab (Random Forest)
повертає інтегральний ризик (0-100 %) та прогноз фактичної тривалості
виконання в днях.
            </p>
        </header>

        <section class="layout">
            <section>
                <form id="task-form">
                    <div class="field">
                        <label for="planned_duration">Планова тривалість,
днів</label>
                        <input id="planned_duration" name="planned_duration"
type="number" min="1" max="60" step="1" value="10" required />
                        <small>Очікувана тривалість задачі за планом.</small>
                    </div>

                    <div class="field">
                        <label for="complexity">Складність задачі (1-5)</label>
                        <input id="complexity" name="complexity" type="number"
min="1" max="5" step="1" value="3" required />
                        <small>1 - проста, 5 - висока алгоритмічна/технічна
складність.</small>
                    </div>

                    <div class="field">
                        <label for="team_experience">Досвід команди (0-
10)</label>
                        <input id="team_experience" name="team_experience"
type="number" min="0" max="10" step="1" value="5" required />
                        <small>Середній рівень досвіду за шкалою 0-10.</small>
                    </div>

                    <div class="field">
                        <label for="priority">Пріоритет задачі (1-3)</label>
                        <input id="priority" name="priority" type="number"
min="1" max="3" step="1" value="2" required />
```

```
                        <small>1 - низький, 3 - критичний для
спринту/релізу.</small>
                    </div>

                    <div class="field">
                        <label for="dependencies">Кількість залежностей</label>
                        <input id="dependencies" name="dependencies"
type="number" min="0" max="10" step="1" value="2" required />
                        <small>Кількість пов’язаних задач або зовнішніх
блокерів.</small>
                    </div>

                    <div class="field">
                        <label for="requirements_clarity">Чіткість вимог (0-
10)</label>
                        <input id="requirements_clarity"
name="requirements_clarity" type="number" min="0" max="10" step="1" value="7"
required />
                        <small>0 - вимоги розмиті, 10 - повністю
формалізовані.</small>
                    </div>

                    <div class="field">
                        <label for="bug_risk">Оціночний ризик дефектів (0-
10)</label>
                        <input id="bug_risk" name="bug_risk" type="number"
min="0" max="10" step="1" value="4" required />
                        <small>Попередня оцінка ймовірності дефектів.</small>
                    </div>

                    <div class="form-footer">
                        <button type="submit" id="submit-btn">
                            Запустити прогноз
                        </button>
                        <span class="status" id="status-text">
                            Дані надсилаються до моделі Random Forest на
бекенді Google Colab.
                        </span>
                    </div>
                </form>
            </section>

            <section>
                <div class="results" id="results-panel">
                    <div class="results-header">
                        <span class="results-title">Результати прогнозу</span>
```

```
                        <span class="results-badge">ML-оцінка в режимі
реального часу</span>
                    </div>

                    <div class="metric">
                        <div class="metric-label">Інтегральний ризик
задачі</div>
                        <div class="risk-bar">
                            <!-- Маска для плавного заповнення -->
                            <div class="risk-fill-mask"></div>
                            <div class="risk-fill" id="risk-fill"></div>
                        </div>
                        <div class="risk-legend">
                            <span>0</span>
                            <span>50</span>
                            <span>100</span>
                        </div>
                        <div class="risk-label-row">
                            <span class="metric-value">
                                <span id="risk-score">–</span> / 100
                            </span>
                            <span id="risk-level-pill" class="risk-level-pill
muted">
                                Рівень ризику буде розраховано
                            </span>
                        </div>
                    </div>

                    <div class="metric">
                        <div class="metric-label">Прогноз фактичної
тривалості</div>
                        <div class="metric-value">
                            ≈ <span id="duration-estimate">–</span> днів
                        </div>
                        <div class="metric-muted">
                            План: <span id="planned-display">–</span>
днів. 
                            Надбавка за ризик відображає запізнення виконання
задачі.
                        </div>
                    </div>

                    <div class="metric">
                        <div class="metric-label">Коротка аналітична
інтерпретація</div>
                        <div class="metric-muted" id="interpretation">
```

```
                            Після першого прогнозу тут буде текстова
інтерпретація рівня ризику та відхилення від плану.
                        </div>
                    </div>
                </div>
            </section>
        </section>
    </main>
</div>

<script>
    const API_URL = "/predict";
    const form = document.getElementById("task-form");
    const submitBtn = document.getElementById("submit-btn");
    const statusText = document.getElementById("status-text");

    const riskScoreEl = document.getElementById("risk-score");
    const riskFillEl = document.getElementById("risk-fill");
    const riskLevelPill = document.getElementById("risk-level-pill");
    const durationEstimateEl = document.getElementById("duration-estimate");
    const plannedDisplayEl = document.getElementById("planned-display");
    const interpretationEl = document.getElementById("interpretation");

    function setLoading(isLoading) {
        if (isLoading) {
            submitBtn.disabled = true;
            submitBtn.textContent = "Обчислення…";
            statusText.textContent = "Виконується запит до моделі на бекенді
Google Colab.";
            statusText.classList.remove("error");
        } else {
            submitBtn.disabled = false;
            submitBtn.textContent = "Запустити прогноз";
        }
    }

    function updateRiskVisualization(score, levelText) {
        const clamped = Math.max(0, Math.min(100, score || 0));
        const ratio = clamped / 100;
        riskFillEl.style.transform = "scaleX(" + (1 - ratio) + ")";

        riskScoreEl.textContent = clamped.toFixed(1);

        let levelClass = "muted";
        if (clamped < 30) {
            levelClass = "low";
```

```
        } else if (clamped < 60) {
            levelClass = "medium";
        } else {
            levelClass = "high";
        }
        riskLevelPill.className = "risk-level-pill " + levelClass;
        riskLevelPill.textContent = levelText;
    }

    function buildInterpretation(riskScore, riskLevel, planned, estimated) {
        if (isNaN(riskScore) || isNaN(planned) || isNaN(estimated)) {
            return "Недостатньо даних для інтерпретації.";
        }
        const delta = estimated - planned;
        const deltaAbs = Math.abs(delta).toFixed(1);
        let deltaText;
        if (delta > 0.5) {
            deltaText = "прогнозується запізнення приблизно на " + deltaAbs + "
дн.";
        } else if (delta < -0.5) {
            deltaText = "прогнозується завершення раніше плану приблизно на " +
deltaAbs + " дн.";
        } else {
            deltaText = "модель очікує виконання в межах планової тривалості.";
        }
        return "Інтегральний ризик складає " + riskScore.toFixed(1) +
            " балів із 100 (рівень: " + riskLevel.toLowerCase() + "). З
урахуванням поєднання складності, залежностей та якості вимог " +
            deltaText;
    }

    form.addEventListener("submit", async function (event) {
        event.preventDefault();

        const formData = new FormData(form);

        const payload = {
            planned_duration: Number(formData.get("planned_duration")),
            complexity: Number(formData.get("complexity")),
            team_experience: Number(formData.get("team_experience")),
            priority: Number(formData.get("priority")),
            dependencies: Number(formData.get("dependencies")),
            requirements_clarity: Number(formData.get("requirements_clarity")),
            bug_risk: Number(formData.get("bug_risk"))
        };
```

```
        setLoading(true);

        try {
            const response = await fetch(API_URL, {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify(payload)
            });

            if (!response.ok) {
                throw new Error("HTTP " + response.status);
            }

            const data = await response.json();
            // Очікуваний формат відповіді бекенду:
            // {
            //   "risk_score": 0-100,
            //   "risk_level": "Низький ризик" | "Середній ризик" | "Високий
ризик",
            //   "estimated_duration_days": число
            // }

            const riskScore = Number(data.risk_score);
            const riskLevel = String(data.risk_level || "");
            const durationEst = Number(data.estimated_duration_days);
            const planned = payload.planned_duration;

            updateRiskVisualization(riskScore, riskLevel);
            durationEstimateEl.textContent = isNaN(durationEst) ? "-" :
durationEst.toFixed(1);
            plannedDisplayEl.textContent = isNaN(planned) ? "-" :
planned.toFixed(1);
            interpretationEl.textContent = buildInterpretation(riskScore,
riskLevel, planned, durationEst);

            statusText.textContent = "Прогноз успішно отримано з бекенду.";
            statusText.classList.remove("error");
        } catch (error) {
            console.error(error);
            statusText.textContent = "Помилка запиту до бекенду. Перевірте
адресу API_URL та запуск сервера в Colab.";
            statusText.classList.add("error");
        } finally {
            setLoading(false);
```

```
        }
    });
</script>
</body>
</html>
```