МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кваліфікаційна наукова праця
на правах рукопису

МИХАЙЛОВ НІКІТА ОЛЕГОВИЧ

УДК 004.6, 004.8, 004.942, 004.021: 519.6

ДИСЕРТАЦІЯ
АЛГОРИТМИ МАШИННОГО НАВЧАННЯ ДЛЯ ПОБУДОВИ ВИСОКОЕФЕКТИВНИХ СИСТЕМ ПРОЄКТНОГО УПРАВЛІННЯ

Спеціальність 124 – Системний аналіз
Галузь знань 12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Н. О. Михайлов
Науковий керівник: Панченко Тарас Володимирович
Кандидат фізико-математичних наук, доцент

КИЇВ – 2025

АНОТАЦІЯ

Михайлов Н.О. Алгоритми машинного навчання для побудови високоефективних систем проєктного управління. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 124 «Системний аналіз». – Київський національний університет імені Тараса Шевченка, Київ, 2025.

Метою роботи є дослідження та розробка методів машинного навчання для створення високоефективних систем проєктного управління та оцінки ризиків проєктів розробки програмного забезпечення. Основною ціллю є досягнення оптимізації процесів створення програм, підвищення якості програмного продукту та зменшення часових і ресурсних витрат.

З ростом завдань та вимог до програмних продуктів стає все важче дотримуватися встановлених графіків, якість кінцевого продукту страждає, а ресурси розробки зазвичай обмежені. Машинне навчання пропонує потенційні рішення для автоматизації процесів управління та аналізу, та є важливим інструментом для підвищення продуктивності та якості розробки програмного забезпечення. Актуальність полягає також у тому, що сучасні розробники програмного забезпечення дедалі більше віддають перевагу штучному інтелекту і машинному навчанню як засобу підвищення ефективності.

Для досягнення мети дослідження в роботі було проведено аналіз сучасних наукових робіт, публікацій, інструкцій та стандартів у галузі управління проєктами, розробки ПЗ і машинного навчання. Це дозволило зрозуміти поточний стан прогресу у цих галузях та виявити існуючі недоліки, які можна вирішити за допомогою нових методів та технік. Для розробки та навчання моделей машинного навчання було зібрано велику кількість даних, що стосуються проєктів розробки ПЗ, включаючи дані про терміни, ресурси, які витрачаються на кожну фазу проєкту, та інші параметри, які впливають на

продуктивність. Дані були проаналізовані та підготовлені для подальшого використання у моделях машинного навчання. На основі зібраних даних були розроблені і навчені моделі машинного навчання, такі як регресійні моделі для передбачення термінів та ресурсів проєкту, класифікатори для ідентифікації ризиків та інші моделі, які відповідають конкретним цілям дослідження. На основі розроблених моделей машинного навчання була створена система підтримки прийняття рішень, яка дозволяє проєктним менеджерам та розробникам аналізувати та оптимізувати процеси управління проєктами та розробки ПЗ. А саме: планування проєктів, розподілення ресурсів і завдань, оцінка ризиків тощо. Результати розроблених моделей та системи були піддані експериментальній перевірці та валідації на реальних проєктах розробки ПЗ. Це допомогло визначити ефективність та точність розроблених методів та засобів. Зібрані дані, результати модельного аналізу та експериментів були оцінені та проаналізовані, щоб зробити висновки щодо досягнутого ефекту та переваг використання методів машинного навчання у системах проєктного управління та аналізу продуктивності розробки програмного забезпечення.

Отримані результати наукового дослідження набули практичного застосування, а розроблені в процесі дослідження моделі, методи та алгоритми показали високу ефективність, про що свідчать відповідні наукові статті за темою дисертації.

Дисертація пропонує наукову новизну через поєднання методів машинного навчання з управлінням проєктами та аналізом продуктивності розробки програмного забезпечення. Основною науковою новизною є розробка імовірнісних моделей для передбачення термінів та ресурсів у проєктах розробки ПЗ, створення системи моніторингу та аналізу продуктивності з використанням аналізу великих обсягів даних та методів машинного навчання. Результати дослідження можуть бути корисними для

промислових компаній та розробників ПЗ для оптимізації їхніх процесів та покращення результатів.

Наукова новизна одержаних результатів:

Вперше:

- Розроблено модель прогнозування тривалості проєктів і ризиків їх затримки та досліджено показники її точності над спеціально підготовленим датасетом.

Удосконалено:

- Раніше розроблені моделі для розв’язання задачі високоефективного планування проєктів.

Основні результати дисертації пройшли апробацію, доповідались на міжнародних конференціях та семінарах. Основні положення, викладені в дисертації, доведено до рівня програмної реалізації та опубліковано у Всеукраїнських виданнях, отже вони підлягають широкому застосуванню як у проведенні подальших досліджень, так і у практичній діяльності. Прикладне значення отриманих результатів полягає в отриманні апробований моделі, втіленій у програмній системі, яка може бути застосована для ефективного планування проєктів, розподілення ресурсів і завдань та оцінки ризиків у режимі реального часу. Результати дисертаційної роботи можуть бути використані для подальших теоретичних та практичних досліджень у сфері інформаційних технологій, у навчальному процесі при підготовці курсів з обробки даних, спеціальних курсів для автоматизації процесів.

Ключові слова: нейронні мережі, машинне навчання, штучний інтелект, системи проєктного управління, продуктивність розробки програмного забезпечення, система підтримки прийняття рішень, планування проєктів,

оцінка ризиків, оптимізація ресурсів, аналіз даних, алгоритми прогнозування, автоматизація управління проєктами, управління ризиками, інтелектуальні системи, адаптивне планування, навчання моделей, рекурентні нейронні мережі, метрики, прийняття рішень, інновації, стратегії, інформаційні технології, комп'ютерна модель, моделювання, сховище даних, штучна нейронна мережа, управління бізнесом, алгоритми обробки даних, тимчасові дані, часова модель, бази даних, структура даних, нечітке значення, задача оптимізації.

ANNOTATION

Mykhailov N.O. Machine Learning Algorithms for Developing High-Efficiency Project Management Systems – A Qualification Scientific Work Manuscript.

Dissertation for the Degree of Doctor of Philosophy in Specialty 124 "System Analysis". – Taras Shevchenko National University of Kyiv, Kyiv, 2025.

This research aims to explore and develop machine learning methods for creating high-efficiency project management systems and assessing risks in software development projects. The primary goal is to optimize the development processes, improve software quality, and reduce time and resource expenditures.

As the complexity and demands for software products increase, adhering to predefined schedules becomes more challenging, the quality of the final product suffers, and development resources are often limited. Machine learning offers potential solutions for automating management and analysis processes, making it a vital tool for improving the productivity and quality of software development. The relevance of the research is reinforced by the growing preference of modern software developers for artificial intelligence and machine learning to enhance efficiency.

To achieve the research objectives, the study includes an analysis of modern scientific works, publications, guidelines, and standards in the fields of project management, software development, and machine learning. This analysis facilitated an understanding of the current progress and revealed existing challenges that could be addressed through new methods and techniques. A significant dataset related to software development projects was collected for developing and training machine learning models. This dataset includes information on timelines, resources expended on each project phase, and other parameters influencing performance. The data was analyzed and prepared for use in machine learning models.

Based on the collected data, machine learning models were developed and

trained, including regression models for predicting project timelines and resources, classifiers for identifying risks, and other models tailored to the research objectives. Using these models, a decision support system was created to enable project managers and developers to analyze and optimize project management and software development processes, such as project planning, resource and task allocation, and risk assessment.

The developed models and systems underwent experimental validation on real software development projects to assess their effectiveness and accuracy. The data collected, along with the results of model analysis and experiments, were evaluated to draw conclusions about the achieved impact and advantages of using machine learning methods in project management systems and performance analysis for software development.

The research findings have practical applications, and the models, methods, and algorithms developed during the study have demonstrated high efficiency, as evidenced by scientific publications on the dissertation topic.

The dissertation contributes scientific novelty by integrating machine learning methods with project management and software development performance analysis. The main scientific novelty lies in developing probabilistic models for predicting timelines and resources in software development projects and creating a monitoring and performance analysis system using big data and machine learning methods. The research outcomes can benefit industrial companies and software developers in optimizing processes and improving results.

The scientific novelty of the results:

For the first time:

- A model for predicting project duration and delay risks was developed and its accuracy was evaluated using a specially prepared dataset.

Improved:

- Previously developed models for solving the task of high-efficiency project planning.

The main results of the dissertation were presented at international conferences and seminars. The key provisions of the dissertation have been implemented into software solutions and published in national journals, making them applicable for further research and practical use. The practical significance of the results is the following: The developed models have been implemented in a software system that can be used for effective project planning, resource and task allocation, and real-time risk assessment. The results of the dissertation can be further utilized for theoretical and practical research in information technology and in educational courses on data processing and process automation.

Keywords: neural networks, machine learning, artificial intelligence, project management systems, software development productivity, decision support system, project planning, risk assessment, resource optimization, data analysis, forecasting algorithms, project management automation, risk management, intelligent systems, adaptive planning, model training, recurrent neural networks, metrics, decision-making, innovation, strategies, information technologies, computer model, modeling, data warehouse, artificial neural network, business management, data processing algorithms, temporal data, temporal model, databases, data structure, fuzzy value, optimization problem.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА:

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Михайлов Н. О. Класифікація користувачів на онлайн-платформах методами машинного навчання // Вісник Київського національного університету імені Тараса Шевченка. Серія: Фізико-математичні науки. 2022. № 4. С. 66-71.
2. Михайлов Н. О. Методи високоефективного планування проєктів: традиційні підходи та машинне навчання // Таврійський науковий вісник. Серія: Технічні науки. 2024. № 4. С. 186-192.
3. Михайлов Н. О. Адаптивна модель планування проєктів та оцінки ризиків із використанням машинного навчання // Науковий вісник Ужгородського університету. Серія: Математика і інформатика. 2024. Т. 45, № 2. С. 216-222.
4. Михайлов Н. О. Проектування та навчання моделі штучного інтелекту для планування і оцінки ризиків проєктів // Таврійський науковий вісник. Серія: Технічні науки. 2024. № 5. С. 124-129.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

1. Михайлов Н. О. Високоефективне планування проєктів: традиційні підходи та машинне навчання // Science, Technology, Innovation: Global Trends and Regional Aspect: матеріали IV Міжнародної науково-практичної конференції (24-27 вересня 2024 р., Таллінн). Таллінн, 2024. С. 42-44.
2. Михайлов Н. О. Система планування проєктів та оцінки ризиків із використанням машинного навчання // Problems of Science Development in the Context of Global Transformations: матеріали V Міжнародної

науково-практичної конференції (1-4 жовтня 2024 р., Загреб). Загреб, 2024. С. 119-120.

3. Михайлов Н. О. Модель штучного інтелекту для планування та оцінки ризиків проєктів // Теоретичні та прикладні аспекти побудови програмних систем: матеріали 15-ї Міжнародної науково-практичної конференції (23-24 грудня 2024 р., Київ). Київ, 2024. С. 44-45.
4. Михайлов Н. О. Розробка моделі штучного інтелекту для планування і оцінки ризиків проєктів // New Ways of Improving Outdated Methods and Technologies: матеріали XVI Міжнародної науково-практичної конференції (17-20 грудня 2024 р., Копенгаген). Копенгаген, 2024. С. 54-55.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Accuracy – точність. Відсоток (або частка) коректно класифікованих прикладів серед усіх перевірених

Adam – Adaptive Moment Estimation, алгоритм адаптивного оцінювання моментів

AI – Artificial Intelligence, штучний інтелект

Anaconda – платформа для створення та управління віртуальними середовищами Python, яка забезпечує зручне керування пакетами та залежностями

API – Application Programming Interface, прикладний програмний інтерфейс. Набір протоколів, підпрограм і засобів для взаємодії між різними програмними компонентами або сервісами

AWS – Amazon Web Services, Хмарна платформа від Amazon для розгортання, масштабування та керування застосунками й сервісами

Azure – Microsoft Azure. Хмарна платформа від Microsoft для створення, розгортання та адміністрування застосунків

B-Tree – “дерево Б”, самобалансована структура даних типу дерево; часто використовується для індексування в реляційних системах управління базами даних (зокрема PostgreSQL)

BRIN – Block Range Index, тип індексу в PostgreSQL, оптимізований для великих таблиць зі згрупованими або відсортованими даними

C4 – Context, Containers, Components, Code, чотирирівнева методика опису архітектури ПЗ (рівні контексту, контейнерів, компонентів і коду)

CNN – Convolutional Neural Network, згорткова нейронна мережа, що використовується для обробки даних із просторовою структурою (наприклад, зображень чи послідовностей).

CPU – Central Processing Unit, центральний процесор

CPM – Critical Path Method, метод критичного шляху

CUDA – Compute Unified Device Architecture, платформа та API від NVIDIA для розробки програмного забезпечення, що використовує GPU для прискорення обчислень

cuDNN – CUDA Deep Neural Network library, бібліотека, оптимізована для прискорення роботи нейронних мереж на GPU за допомогою CUDA

Dense – повнозв'язаний шар нейронної мережі, де кожен нейрон з'єднаний з усіма нейронами попереднього шару; використовується для виявлення складних залежностей у даних

ELU – Exponential Linear Unit, експоненціальна лінійна одиниця, функція активації

EF – Early Finish, раннє завершення. Найближчий (найраніший) можливий час завершення завдання (у методі CPM).

ES – Early Start, ранній початок. Найближчий (найраніший) можливий час початку завдання (у методі CPM).

F1-score – гармонійне середнє між Precision та Recall, що свідчить про збалансованість точності й повноти.

Float – резерв часу в CPM. Запас часу, на який можна відкласти завдання, не впливаючи на загальний час проєкту.

GIN – Generalized Inverted Index, тип індексу в PostgreSQL для ефективного пошуку в масивах, JSON і повнотекстового пошуку.

GiST – Generalized Search Tree, Гнучка структура індексування у PostgreSQL (і не лише), застосовується для просторових пошуків, R-дерев та ін.

Git – система контролю версій, що використовується для документування змін та забезпечення повторюваності процесу навчання

GPU – Graphics Processing Unit, графічний процесор

HTTP – HyperText Transfer Protocol

JIRA – система керування проєктами й завданнями від Atlassian.

JQL – Jira Query Language, мова запитів у JIRA для пошуку та фільтрації завдань.

JSON – JavaScript Object Notation

JWT – JSON Web Token

Kaggle – платформа для змагань із науки про дані, обміну наборами даних і навчальними матеріалами.

Keras – Бібліотека Python для швидкого створення і навчання нейронних мереж

LeakyReLU – Leaky Rectified Linear Unit, модифікована версія функції активації ReLU

LF – Late Finish, пізнє завершення. Найпізніший час, коли завдання може бути завершене (у CPM), не затримуючи проєкт

LS – Late Start, пізній початок. Найпізніший час, коли завдання може розпочатися (у CPM), щоб не вплинути на загальні терміни.

MAE – Mean Absolute Error, середня абсолютна похибка. Середній модуль різниці між прогнозованим і фактичним значенням.

MAPE – Mean Absolute Percentage Error, середня абсолютна відсоткова похибка. Показник, що відображає відсоткове відхилення прогнозу від реального значення у середньому.

ML – Machine Learning, машинне навчання

MSE – Mean Squared Error, середньоквадратична похибка. Усереднений квадрат різниці між прогнозом і фактичним значенням.

NumPy – Numerical Python. Пакет Python для оперування багатовимірними масивами й виконання наукових обчислень.

OAS – OpenAPI Specification. Стандарт опису RESTful API, що забезпечує документування та інтеграцію сервісів.

One-Hot Encoding – Метод перетворення категорійних ознак у набір бінарних (0/1) індикаторів для навчання ML-моделей.

Pandas – Python Data Analysis. Бібліотека Python для обробки табличних даних, їх аналізу та підготовки.

PERT – Program Evaluation and Review Technique

Precision – влучність. Частка справді релевантних об’єктів серед усіх, які модель позначила як «позитивні» (у класифікації).

RDBMS – Relational Database Management System, система управління реляційними базами даних.

Recall – повнота. Частка релевантних зразків, які класифікатор позначив як “позитивні”, серед усіх справді позитивних зразків

ReLU – Rectified Linear Unit

REST – Representational State Transfer

RMSprop – Root Mean Square Propagation, алгоритм оптимізації, який динамічно змінює швидкість навчання для параметрів, використовуючи середні квадрати градієнтів

RMSE – Root Mean Squared Error, середньоквадратична похибки

Scikit-learn – основна бібліотека Python для класичних алгоритмів ML (класифікація, регресія, кластеризація тощо)

SGD – Stochastic Gradient Descent, стохастичний градієнтний спуск

Spring Boot – фреймворк на мові Java для спрощеного створення мікросервісів і вебзастосунків

SVM – Support Vector Machine, метод опорних векторів

Swagger – Інструмент і специфікація для документування і тестування REST API, надає вебінтерфейс для HTTP-запитів

TensorFlow – бібліотека машинного навчання від Google, підтримує обчислення на CPU, GPU, TPU; застосовується для глибинних нейронних мереж

TPU – Tensor Processing Unit, пеціалізований процесор від Google, оптимізований для швидкого навчання великих нейронних мереж.

UI – User Interface, користувацький інтерфейс
Засіб взаємодії людини з програмним забезпеченням.

UUID – Universally Unique Identifier, унікальний ідентифікатор об’єктів у розподілених системах.

YAML – YAML Ain't Markup Language. Формат даних, який широко використовується для опису конфігураційних файлів та специфікацій API.

ВСТУП

Актуальність теми. Сучасні проєкти, особливо в галузі розробки програмного забезпечення, стають все складнішими і більш об'ємними. Управління такими проєктами вимагає ефективних інструментів і методів, щоб забезпечити їх успішне завершення. Підприємства шукають способи оптимізувати свої процеси розробки та управління проєктами, щоб збільшити продуктивність, знизити витрати та покращити якість кінцевого продукту. Завдяки технологічному прогресу сьогодні доступно величезна кількість даних, які можуть бути використані для аналізу і прийняття рішень. Машинне навчання надає засоби для ефективного аналізу цих даних та використання їх для підтримки управлінських рішень. Штучний інтелект і машинне навчання здобувають все більшу популярність у бізнесі та індустрії як засоби для автоматизації процесів, вирішення складних завдань і забезпечення конкурентних переваг. Машинне навчання може забезпечити засоби для створення систем, які можуть швидко аналізувати великі обсяги даних та надавати рекомендації для управління проєктами.

Метою роботи є дослідження та розробка методів машинного навчання для створення високоефективних систем проєктного управління та аналізу продуктивності та оцінки ризиків розробки програмного забезпечення.

Для досягнення вказаної мети у дисертації поставлено наступні **наукові завдання**:

- виконати порівняльний аналіз існуючих систем проєктного управління;
- визначити основні недоліки розроблених алгоритмів ефективного планування проєктів та ідентифікувати шляхи покращення;
- визначити критерії точності, за якими розробити та оптимізувати побудовані моделі та алгоритми високоефективних систем проєктного управління;

- розробити ефективні за визначеними критеріями модель та алгоритм проєктного управління;
- дослідити показники точності та ефективності запропонованих моделі та алгоритму.

Об’єктом дослідження є системи проєктного управління.

Предметом дослідження є моделі, методи, алгоритми та інформаційні технології для побудови систем ефективного планування проєктів, розподілення ресурсів і завдань та оцінки ризиків у режимі реального часу.

Методи дослідження. При виконанні поставлених завдань використовувались: методи та засади системного аналізу, інформаційних технологій та комп’ютерних наук; методи попередньої обробки інформації та токенізації; методи та підходи до розробки програмного забезпечення; методи машинного навчання, зокрема нейронні мережі.

Наукова новизна отриманих результатів. Наукова новизна одержаних результатів полягає в розробці алгоритму та математичної моделі для побудови високоефективних систем проєктного управління.

Вперше:

- Розроблено модель прогнозування тривалості проєктів і ризиків їх затримки та досліджено показники її точності над спеціально підготовленим датасетом.

Удосконалено:

- Раніше розроблені моделі для розв’язання задачі високоефективного планування проєктів.

Наукове та практичне значення роботи. Результати дисертації отримані при розробці алгоритму підтвердили точність та ефективність

розробленої моделі та для розв’язання комплексної науково-прикладної проблеми побудови високоефективних систем проєктного управління.

Запропонований алгоритм для планування проєктів на основі машинного навчання не лише прогнозує тривалість завдань, але й оцінює ризики, що дає змогу проєктним командам реагувати на можливі проблеми до їх виникнення. Завдяки своїй здатності працювати навіть із невеликими наборами даних, алгоритм є гнучким і придатним для багатьох типів проєктів, незалежно від їх масштабу. Навіть якщо дані обмежені, модель може ефективно навчатися та генерувати корисні прогнози, що є значною перевагою в умовах, коли доступність історичних даних мінімальна. Однак, із розширенням обсягів даних точність прогнозів і оцінка ризиків значно покращуються.

Підтвердженням практичної важливості результатів дисертації є довідка про впровадження в Codillas GmbH (довідка № 10 від 10.03.2025 р.).

Зв’язок роботи з науковими програмами, планами, темами, грантами. Дисертаційна робота є складовою частиною наукових робіт, які були проведені на кафедрі теорії та технології програмування факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка при виконанні фундаментальної теми «Теорія і методи розробки інтелектуальних інформаційних технологій та систем» (№16КФ015-02, номер держреєстрації 0116U006378, 0116U004780).

Особистий внесок здобувача. Дисертація є самостійною науковою працею, в якій висвітлені власні розробки та ідеї автора, які дозволили вирішити поставлені завдання. Робота містить теоретичні положення і висновки, особисто сформульовані дисертантом. Використані в дисертації ідеї, гіпотези або положення інших авторів мають відповідні посилання та використані виключно для підкріплення ідей здобувача. За темою дисертації було опубліковано 4 наукових праці.

Апробація матеріалів дисертації. Основні положення і висновки

дисертаційного дослідження були обговорені на наукових семінарах кафедри теорії та технології програмування факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка і отримали схвальні відгуки.

Результати дисертаційного дослідження пройшли апробацію, їх оприлюднено у доповідях міжнародних та всеукраїнських наукових конференцій, семінарів:

- Михайлов Н. О. Методи високоефективного планування проєктів: традиційні підходи та машинне навчання // Science, Technology, Innovation: Global Trends and Regional Aspect: матеріали IV Міжнародної науково-практичної конференції (24-27 вересня 2024 р., Таллінн). Таллінн, 2024. С. 42-44.

- Михайлов Н. О. Адаптивна модель планування проєктів та оцінки ризиків із використанням машинного навчання // Problems of Science Development in the Context of Global Transformations: матеріали V Міжнародної науково-практичної конференції (1-4 жовтня 2024 р., Загреб). Загреб, 2024. С. 119-120.

- Михайлов Н. О. Модель штучного інтелекту для планування та оцінки ризиків проєктів // Теоретичні та прикладні аспекти побудови програмних систем: матеріали 15-ї Міжнародної науково-практичної конференції (23-24 грудня 2024 р., Київ). Київ, 2024. С. 44-45.

- Михайлов Н. О. Проєктування та навчання моделі штучного інтелекту для планування і оцінки ризиків проєктів // New Ways of Improving Outdated Methods and Technologies: матеріали XVI Міжнародної науково-практичної конференції (17-20 грудня 2024 р., Копенгаген). Копенгаген, 2024. С. 54-55.

Публікації. Результати дисертації було опубліковано у 4 наукових працях в наукових журналах та збірниках наукових праць, які індексується в наукометричних базах Scopus або Web of Science.

Структура та обсяг дисертації. Дисертація містить: анотацію, вступ, 4 розділи, висновки, список використаних джерел. Обсяг дисертації – 136 сторінок, основної частини – 102 сторінки. Робота містить 9 таблиць, 33 рисунки, список використаних джерел з 144 найменувань.

РОЗДІЛ 1. ЗАДАЧА ВИСОКОЕФЕКТИВНОГО ПЛАНУВАННЯ ПРОЄКТІВ

В сучасному світі, де швидкість і ефективність грають ключову роль у досягненні успіху, високоефективне планування проєктів стає необхідністю для організацій будь-якої сфери діяльності. Задача високоефективного планування проєктів полягає в розробці та впровадженні стратегічних і тактичних кроків з метою оптимізації використання ресурсів, зменшення часових витрат і підвищення якості результату. Це процес, який вимагає систематичного підходу, глибокого аналізу поточної ситуації, врахування ризиків та управління змінами. Високоефективне планування проєктів дозволяє забезпечити успішне виконання завдань, досягнення поставлених цілей і забезпечення конкурентних переваг на ринку. У цьому контексті важливо розглядати сучасні методи і інструменти управління проєктами, використання технологій штучного інтелекту та машинного навчання для автоматизації процесів та підвищення ефективності планування [35, 41].

1.1. Постановка задачі

Планування проєктів в сучасних умовах вимагає вдосконалення стратегій та методів для забезпечення високої ефективності та успішності реалізації. Однак, існуючі методи часто не враховують всіх аспектів проєкту та не забезпечують оптимального використання ресурсів. Тому постановка задачі полягає у розробці та застосуванні алгоритмів та стратегій високоефективного планування проєктів, які забезпечують найбільш оптимальне розподілення ресурсів, мінімізацію витрат часу та зусиль, а також максимізацію якості та результативності проєкту в цілому [60]. Це включає розробку нових методик аналізу та прогнозування, використання інструментів машинного навчання для оптимізації процесів управління та розробки програмного забезпечення, а також створення систем підтримки прийняття рішень на основі аналізу

великих обсягів даних. Основна мета полягає у досягненні ефективного та результативного управління проєктами, що сприятиме підвищенню продуктивності, зменшенню ризиків та досягненню поставлених цілей [30, 141].

1.2. Необхідність ефективного розподілення ресурсів та сценарії застосування результатів роботи

Ефективне розподілення ресурсів є ключовим аспектом успішного управління проєктами у сучасних умовах. Зростання складності та обсягів проєктів, швидкі темпи змін у наукових і бізнес середовищах та обмежені ресурси вимагають розробки стратегій, які дозволяють максимально ефективно використовувати наявні ресурси. Ефективне розподілення ресурсів дозволить підвищити продуктивність роботи команди, знизити час та витрати на реалізацію проєкту, а також забезпечити високу якість результату [16, 45].

Сценарії застосування результатів роботи

1. Промислові компанії та IT-сектор: розроблені методи та алгоритми ефективного планування проєктів можуть бути використані промисловими компаніями та IT-підприємствами для оптимізації управління проєктами різної складності та масштабу. вони допоможуть забезпечити вчасне завершення проєктів, підвищити ефективність використання ресурсів та покращити якість продукції [39].
2. Навчальні заклади: результати дослідження можуть бути використані у навчальному процесі для підготовки фахівців у галузі управління проєктами та інформаційних технологій. вони допоможуть студентам отримати практичні навички з використання сучасних методів та інструментів управління проєктами.

3. Дослідницькі проєкти: результати дослідження можуть бути використані для подальших досліджень у галузі управління проєктами та розробки програмного забезпечення. Це відкриває нові можливості для розвитку та вдосконалення методів управління проєктами та прогнозування результатів [33].
4. Інноваційні стартапи: стартапи можуть скористатися розробленими методами для оптимізації управління власними проєктами, що допоможе їм ефективніше впроваджувати ідеї та продукти на ринок [59].
5. Урядові організації та громадські ініціативи: результати дослідження можуть бути використані урядовими організаціями та громадськими ініціативами для планування та управління проєктами в різних галузях, включаючи соціальні та інфраструктурні ініціативи [11].

1.2.1. Значення для проєктного менеджменту

Ефективне планування проєктів є критично важливим етапом у життєвому циклі будь-якого проєкту. Результати досліджень у цій області мають вирішальне значення для практики проєктного менеджменту, оскільки вони сприяють досягненню ряду ключових цілей та завдань, що стоять перед керівниками проєктів [40].

Ретельне аналізування даних та розробка прогностичних моделей дозволяють проєктним менеджерам оптимізувати ресурси та час, що витрачаються на кожну фазу проєкту. Це дозволяє підвищити продуктивність команди, зменшити час виконання проєкту та знизити загальні витрати [20].

Аналіз та обробка великих обсягів даних дозволяє визначати оптимальні стратегії розподілу ресурсів на різних етапах проєкту. Застосування математичних моделей і алгоритмів оптимізації дозволяє ефективно використовувати обмежені ресурси, мінімізуючи при цьому час і витрати. Одним з важливих аспектів є виявлення та управління ризиками. Аналіз і

прогнозування ризиків за допомогою статистичних методів та моделей дозволяє передбачити можливі проблеми та вжити необхідні заходи для їхнього уникнення або зменшення впливу. Додатково, розробка систем підтримки прийняття рішень на основі аналізу даних дозволяє керівникам проєктів приймати обґрунтовані рішення, підкріплені об'єктивними даними та прогностичними моделями. Це робить процес управління проєктами більш прогнозованим та ефективним [19, 42].

У світлі цього, результати досліджень у сфері ефективного планування проєктів стають критично важливими для підвищення конкурентоспроможності організацій та забезпечення успішного виконання проєктів у сучасних умовах ринкової конкуренції.

1.2.2. Оцінка ризиків в реальному часі

Оцінка ризиків в реальному часі важлива з кількох причин. По-перше, це дозволяє забезпечити безперервний моніторинг та контроль за станом проєкту, що допомагає уникнути негативних наслідків, таких як затримки, перевитрати або невдачі. По-друге, це дозволяє швидко реагувати на зміни в умовах проєкту та приймати вчасні заходи для мінімізації ризиків та використання можливостей. По-третє, це сприяє підвищенню ефективності та продуктивності управління проєктами, що веде до досягнення кращих результатів та забезпечує успішне завершення проєкту [77, 130].

Існують різні механізми для оцінки ризиків в реальному часі, включаючи використання спеціалізованих програмних систем та систем внутрішнього контролю та звітності, а також алгоритмів та нейромереж [95].

Спеціалізовані програмні системи та системи внутрішнього контролю та звітності часто використовуються організаціями для оцінки ризиків у реальному часі. Ці системи мають декілька переваг, таких як легкість використання та автоматизація процесу аналізу даних. Вони забезпечують

можливість створювати детальні звіти про стан проєкту та ідентифікувати потенційні ризики, що дозволяє оперативно реагувати на них [43, 112].

Однак ці системи можуть мати свої обмеження, такі як потреба в одноразовому налаштуванні та обмеженості в аналізі. Вони можуть виявитися не такими ефективними у прогнозуванні складних ризиків або виявленні складних залежностей між різними факторами. У порівнянні з цим, алгоритми та нейромережі можуть бути більш ефективними у виявленні та прогнозуванні ризиків у реальному часі [72]. Вони відрізняються вищою гнучкістю та прогностичною силою, що дозволяє їм адаптуватися до різних типів даних та складних сценаріїв. Алгоритми та нейромережі можуть автоматично аналізувати та обробляти великі обсяги даних в реальному часі, що дозволяє оперативно виявляти та реагувати на потенційні ризики [28, 140].

Хоча обидва підходи мають свої переваги та обмеження, використання алгоритмів та нейромереж для оцінки ризиків у реальному часі може бути ефективним рішенням для забезпечення безпеки та стабільності виконання проєктів.

1.3. Відомі підходи до вирішення задачі

Існує кілька відомих підходів до вирішення задачі високоефективного планування проєктів, кожен з яких має свої особливості та переваги.

1.3.1. Метод критичного шляху

Метод критичного шляху (Critical Path Method, CPM) є методом управлення проєктами, який застосовується для визначення критичних завдань, що безпосередньо впливають на мінімальний час реалізації всього проєкту. CPM передбачає побудову графіка проєкту, в якому кожне завдання має чітко визначені часові межі. Основною ідеєю цього методу є визначення тієї послідовності задач, яка встановлює загальну тривалість проєкту. Таким

чином, затримка будь-якого завдання, що перебуває на критичному шляху, автоматично спричиняє збільшення термінів реалізації всього проєкту [68].

Процедура методу критичного шляху охоплює кілька ключових етапів. Спершу формується детальний перелік усіх завдань проєкту, кожне з яких чітко визначається і включається до загального списку. Наступним кроком є встановлення залежностей між завданнями, що передбачає визначення порядку, в якому завдання мають бути виконані. Після цього здійснюється оцінювання тривалості кожного завдання. Потім створюється сіткова діаграма, яка візуалізує всі завдання та їхні взаємозв'язки у вигляді графа: вузли позначають окремі завдання, а стрілки – залежності між ними. Останнім кроком є визначення критичного шляху, який відповідає найдовшій послідовності завдань і має вирішальне значення для встановлення загального терміну виконання проєкту [2, 65].

CPM використовує кілька ключових параметрів:

- (ES) ранній початок — найближчий час, коли завдання може бути розпочато.
- (EF) раннє завершення — найближчий час завершення завдання, який розраховується як

$$EF = ES + t,$$

де *t* — це тривалість виконання завдання.

- (LS) пізній початок — найпізніший можливий час початку завдання.
- (LF) пізнє завершення — найпізніший час завершення завдання, розраховується як:

$$LS = LF - t.$$

Резерв часу (Float) відображає період, на який можна затримати виконання певного завдання, не впливаючи при цьому на загальний термін завершення проєкту. Він визначається за формулою:

$$Float = LS - ES.$$

Якщо значення резерву часу дорівнює нулю, це означає, що відповідне завдання належить до критичного шляху проєкту.

Метод критичного шляху характеризується низкою вагомих переваг. Він забезпечує чітке визначення критичних завдань, які безпосередньо впливають на терміни успішного завершення проєкту, що дозволяє менеджерам зосередити увагу саме на цих елементах. CPM сприяє оптимальному розподілу часу та ресурсів, що підвищує загальну ефективність виконання проєкту. Ще однією перевагою методу є можливість заздалегідь виявляти потенційні затримки та своєчасно реагувати на ризики. Водночас метод має певні обмеження: він недостатньо гнучкий для проєктів із динамічними умовами, оскільки передбачає детальне планування усіх завдань на початковій стадії. Крім того, CPM може бути важко застосовувати для проєктів з високим рівнем невизначеності, що часто вимагає регулярних коригувань плану [64, 118].

Метод критичного шляху широко використовується в інженерних, будівельних та IT-проєктах, у яких дотримання термінів є ключовим чинником успішності. CPM дає змогу на ранніх стадіях ідентифікувати критичні завдання та ефективно розподілити наявні ресурси для забезпечення своєчасного завершення проєкту [2, 113].

1.3.2. Метод оцінки та аналізу

Метод оцінки та аналізу програм (Program Evaluation and Review Technique, PERT) застосовується для аналізу та прогнозування часу, необхідного для виконання завдань у межах проєкту. Метод було створено у

1950-х роках для потреб військово-морських сил США з метою ефективного управління проєктами, що характеризуються значною невизначеністю. Основною метою PERT є визначення ймовірних часових інтервалів для кожного окремого завдання, що дозволяє менеджерам точніше оцінити загальну тривалість проєкту. Для цього PERT передбачає застосування трьох типів оцінок часу виконання завдання:

- (*O*) оптимістична оцінка — мінімальний час для завершення завдання,
- (*P*) песимістична оцінка — максимальний час виконання завдання,
- (*M*) ймовірна оцінка — найбільш вірогідний час виконання завдання.

На основі цих експертних оцінок — оптимістичної, песимістичної та найбільш ймовірної — у класичному підході PERT очікувану тривалість завдання обчислюють за формулою:

$$T_e = \frac{(O + 4M + P)}{6},$$

де значення M враховується з більшою вагою, оскільки вважається найбільш ймовірним сценарієм. Така формула є евристичним наближенням середнього значення за припущенням, що розподіл тривалості має одну моду - унімодальний і є приблизно симетричним або слабко асиметричним [2, 114].

Водночас, цей підхід має суттєві обмеження. Зокрема, у випадках, коли розподіл тривалості є мультимодальним, тобто має декілька локальних максимумів — кілька найбільш ймовірних значень, застосування формули втрачає статистичну коректність. У таких ситуаціях значення M не може бути однозначно визначеним, і формула перестає відображати реальну структуру ймовірностей. Крім того, метод не враховує форму розподілу, варіації або наявність асиметрій за межами трьох точок (O, M, P), що ще більше обмежує його точність.

Для кращого розуміння обмежень формули PERT розглянемо ситуацію з

мультимодальним розподілом, тобто таким, що має декілька локальних максимумів. Наприклад, якщо для певного завдання існує два характерних сценарії виконання: один — швидкий (за наявності досвідченої команди), інший — повільний (за потреби в донавчанні персоналу), обидва можуть мати високу ймовірність. У такому випадку розподіл тривалості матиме дві моди, і жодне значення *M* не зможе однозначно представляти найімовірніший сценарій [79]. Використання середнього зваженого значення, як у формулі PERT, призведе до втрати точності, оскільки воно фактично усереднить між двома реальними, відмінними один від одного сценаріями. Графічно це можна зобразити як розподіл з двома піками — на відміну від класичного нормального розподілу, який має лише один.

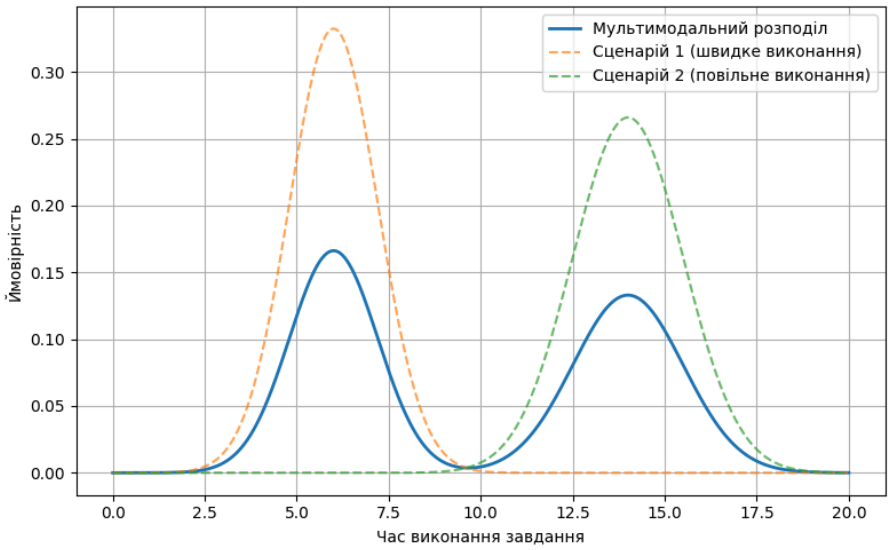

Рис 1. Приклад мультимодального розподілу тривалості завдання

У таких випадках замість спрощеної формули варто використовувати методи, здатні оперувати повним розподілом і враховувати кілька сценаріїв — зокрема, імітаційне моделювання Монте-Карло, яке дозволяє моделювати тисячі варіантів розвитку подій на основі заданих ймовірностей та розподілів.

Однією з переваг методу оцінки та аналізу програм є врахування невизначенності у плануванні часу для виконання задач. Це робить метод

PERT особливо ефективним у випадку масштабних і комплексних проєктів, де точне визначення часових рамок виконання завдань заздалегідь є ускладненим. Окрім того, застосування цього методу забезпечує кількісну оцінку ризиків, що сприяє більш ефективному управлінню непередбачуваними ситуаціями. Водночас, використання PERT вимагає значних зусиль для збору та аналізу даних щодо тривалості кожного завдання, що може виявитися досить трудомістким. Крім того, метод є надмірно складним для простих або невеликих проєктів, де така деталізація зазвичай є недоцільною.

Даний метод може використовується у комбінації з методом критичного шляху, що дозволяє поєднувати точність CPM із гнучкістю PERT у складних та невизначених умовах [2, 66].

1.3.3. Метод дерева рішень

Метод дерева рішень (Decision Tree Method) застосовується в умовах, коли існує кілька можливих сценаріїв розвитку подій. Він дає змогу візуалізувати процес прийняття рішень у вигляді графічної моделі, що відображає альтернативні варіанти дій, ймовірні події та їх можливі наслідки.

Дерево рішень є графічною моделлю процесу ухвалення рішень, де кожен етап представлений вузлами двох основних типів: вузлами рішень, у яких обирають одну з кількох можливих дій, та вузлами випадкових подій, що описують наслідки з певними ймовірностями. Гілки, які відходять від вузла рішення, відповідають потенційним діям, а гілки від вузла випадкової події — сценаріям, що можуть статися з визначеною імовірністю. Завершальні вузли відображають корисність чи вартість, що виникають у результаті конкретної послідовності рішень і випадкових подій. Така структура дозволяє систематизувати всі можливі сценарії та, порівнюючи їх за обраним критерієм, визначити оптимальний стратегічний шлях [2, 84].

Кожна гілка, що виходить із вузла рішення, відповідає одному з варіантів

дії, а гілки від вузла випадкових подій — можливим наслідкам з їхніми ймовірностями. У фінальних (термінальних) вузлах зазначають корисність або витрати, що виникають у результаті конкретної послідовності рішень і подій. Дана структура дозволяє охопити весь спектр можливих сценаріїв і проаналізувати очікувані результати для кожного з них. Це, у свою чергу, сприяє систематизації інформації та прийняттю обґрунтованих рішень на всіх етапах проєкту, допомагаючи менеджерам обирати найбільш ефективні варіанти дій у складних і непередбачуваних умовах.

З математичної точки зору цей метод можна формалізувати як задачу максимізації очікуваної корисності:

$$EU(a) = \sum_{i=1}^{n} P_i \times U_i,$$ де:

- *Ui* — корисність результату,
- *Pi* — ймовірність настання результату *i*,
- *EU(a)* — очікувана корисність рішення *a*.

Дана формула відображає очікувану корисність рішення *a* за умов одного рівня ймовірнісного розгалуження. Проте в реальних задачах дерево рішень зазвичай містить кілька послідовних кроків, де вузли рішень чергуються з вузлами випадкових подій, а кожен із них може утворювати додаткові гілки. У такому разі очікувана корисність для всієї структури обчислюється рекурсивно, починаючи від кінцевих вузлів і рухаючись до початкового: у вузлах випадкових подій підсумовують корисності наступних вузлів з урахуванням їхніх імовірностей, а у вузлах рішень із кількох альтернатив обирають ту, що дає максимальну очікувану корисність. Така процедура поєднує операції максимізації у вузлах рішень та підсумовування у вузлах випадковостей, унаслідок чого формула повної ймовірності є лише

одним зі складових компонентів аналізу. Для розрахунку очікуваної корисності в багаторівневих деревах рішень застосовують рекурсивний підхід. Нехай $V(n)$ позначає очікувану корисність у вузлі n. Тоді:

$$V(n) = \begin{cases} U(n), & \text{якщо } n \text{ є кінцевим вузлом,} \\ \max_{c \in \mathcal{C}(n)} V(c), & \text{якщо } n \text{ є вузлом рішення,} \\ \sum_{c \in \mathcal{C}(n)} p_{n \to c} V(c), & \text{якщо } n \text{ є вузлом випадкових подій.} \end{cases}$$

- $U(n)$ — корисність у термінальному вузлі n,
- $C(n)$— множина вузлів, на які можна перейти з вузла n,
- $p_{n \to c}$ — імовірність переходу з n до c.

Процес побудови дерева рішень розпочинається з визначення початкової задачі, яка виступає в ролі кореневого вузла дерева. На наступному етапі формулюються альтернативні варіанти дій, кожен з яких утворює нові вузли, що репрезентують потенційні рішення або наслідки. Кожна гілка дерева супроводжується ймовірнісною оцінкою події та відповідним результатом, який може виникнути внаслідок вибору певного варіанта. Завершується процес вибором оптимального рішення, що ґрунтується на порівнянні очікуваних корисностей для кожного з можливих сценаріїв [102].

Метод дерева рішень забезпечує ефективну візуалізацію складних процесів прийняття рішень, дозволяючи чітко оцінити наслідки кожного можливого варіанту. Завдяки цьому користувачі мають змогу краще розуміти логіку вибору та обґрунтовувати управлінські рішення. Крім того, цей підхід дає можливість кількісно оцінити ризики та ймовірності для кожного потенційного результату. Його простота й наочність роблять метод зручним і доступним для широкого кола фахівців. Водночас, за наявності великої кількості альтернатив або змінних, дерево рішень може стати надто

громіздким і складним у застосуванні. До того ж, оцінки ймовірностей і корисностей часто основуються на суб'єктивних припущеннях, що може знижувати точність і надійність результатів аналізу [97].

У сфері управління проєктами метод дерева рішень є корисним інструментом для прийняття обґрунтованих рішень щодо розподілу ресурсів, оцінки ризиків і вибору оптимальних стратегій для досягнення поставлених цілей. Зокрема, у ситуаціях, коли потрібно обрати технологію реалізації проєкту або визначити напрям розвитку, дерева рішень дозволяють порівняти можливі варіанти, проаналізувати їхні наслідки та прийняти рішення на основі прогнозованих результатів і пов'язаних із ними ризиків [2, 89].

1.3.4. Ітеративний та інкрементальний підхід

Ітеративно-інкрементальний підхід об'єднує принципи повторюваного виконання (ітеративність) і поступового нарощування функціональності (інкрементальність) у процесі реалізації проєкту. Такий підхід є особливо ефективним у випадках, коли вимоги до проєкту змінюються або не можуть бути чітко сформульовані на початковому етапі. Він широко використовується в галузі програмної інженерії, а також в управлінні проєктами, що орієнтовані на інновації, де необхідна гнучкість, адаптивність і постійне вдосконалення продукту [34, 69].

Ітеративність передбачає поділ проєкту на низку коротких циклів, протягом яких команда займається розробкою, тестуванням і вдосконаленням продукту або його окремих елементів. Після завершення кожної ітерації результати оцінюються, а подальші дії коригуються відповідно до зворотного зв'язку від зацікавлених сторін (стейкхолдерів). Інкрементальність, у свою чергу, означає поступове розширення функціональності продукту: на кожному етапі до нього додаються нові функції чи компоненти, що підвищує його загальну цінність і забезпечує зростання функціональних можливостей у міру

просування проєкту [6].

Продуктивність ітеративного підходу зазвичай оцінюється за допомогою показника швидкості (*Velocity*), який відображає обсяг виконаних завдань або реалізованих одиниць функціональності протягом однієї ітерації.:

$$Velocity = \frac{\text{Завдання виконані за ітерацію}}{\text{Час на ітерацію}}$$

Цей показник дозволяє прогнозувати, який обсяг роботи команда зможе виконати в наступних ітераціях [2, 9].

Регулярне тестування та внесення змін на кожному етапі реалізації проєкту сприяють зниженню ризиків, оскільки дозволяють своєчасно виявляти й усувати недоліки, тим самим зменшуючи ймовірність виникнення критичних помилок у фінальній версії продукту.

Ітеративно-інкрементальний підхід широко використовується в рамках гнучких методологій управління проєктами, таких як Agile та Scrum, а також у процесах розробки програмного забезпечення. Завдяки цьому підходу можливо отримувати функціональні версії продукту вже на ранніх стадіях проєкту, що забезпечує швидке отримання зворотного зв’язку та дає змогу поступово вдосконалювати продукт відповідно до змін у вимогах або умовах реалізації проєкту [8, 46].

1.3.5. Використання алгоритмів машинного навчання

Застосування алгоритмів машинного навчання в управлінні проєктами набуває все більшої популярності завдяки їхній здатності ефективно опрацьовувати великі обсяги даних, виявляти приховані закономірності та здійснювати точні прогнози. Технології машинного навчання (ML) активно впроваджуються для оптимізації процесів планування, прогнозування строків виконання, управління ризиками та ефективного розподілу ресурсів у рамках складних і багатокомпонентних проєктів [2, 52].

У дослідженні “A Hybrid Deep Learning Approach to Integrate Predictive Maintenance Planning” запропоновано гібридну модель, яка поєднує згорткові нейронні мережі (CNN) із механізмами уваги (Attention Mechanism) [104]. Цей підхід дозволяє інтегрувати тактичне виробниче планування та прогнозне технічне обслуговування, значно підвищуючи точність прогнозів.

Стаття “Unraveling the Complexity: A Comparative Analysis of Neural Network Topology and Architecture for Project Planning” проводить детальний порівняльний аналіз різних архітектур нейронних мереж [52]. Дослідники акцентують увагу на критичному значенні вибору відповідної структури мережі для досягнення оптимальних результатів у плануванні проєктів.

Важливість інтеграції ансамблевих методів показана в роботі “Software Risk Prediction at Requirement and Design Phase: An Ensemble Machine Learning Approach” [18]. Автори демонструють, що комбінування декількох моделей машинного навчання дозволяє отримати стабільніші та більш надійні прогнози ризиків на ранніх стадіях життєвого циклу програмних проєктів.

Дослідження “Predictive Project Management: Enhance Efficiency” обговорює застосування передбачувальної аналітики та глибокого навчання для покращення ефективності управління проєктами [17]. Автори роблять висновок, що такі технології дозволяють менеджерам проєктів ухвалювати більш обґрунтовані рішення, базовані на точних прогнозах.

Машинне навчання дозволяє здійснювати прогнозування тривалості як окремих завдань, так і всього проєкту на основі історичних даних про попередні проєкти. Для реалізації таких прогнозів застосовуються різноманітні моделі, зокрема лінійна регресія, алгоритми градієнтного бустингу та нейронні мережі. Ці підходи дають змогу виявити залежності між характеристиками завдань і їхньою фактичною тривалістю, що дозволяє точніше оцінити терміни виконання майбутніх робіт.

У випадку лінійної регресії для прогнозування тривалості може бути використана наступна формула:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$, де

- $\hat{y}$ — прогнозований час виконання завдання,
- β_0 — вільний член,
- $\beta_1, \beta_2, \ldots \beta_n$ — вагові коефіцієнти відповідних характеристик завдання $x_1, x_2, x_3 \ldots x_n$.

Набір ознак x_i , які використовуються в моделі, не є фіксованим або універсальним. Для різних класів завдань релевантні характеристики можуть суттєво відрізнятися — наприклад, для програмування важливими можуть бути мова, розмір коду, кількість взаємодіючих модулів, тоді як для будівництва — тип конструкції, доступність ресурсів, погодні умови тощо. Визначення релевантних ознак зазвичай здійснюється емпірично, шляхом аналізу даних у межах конкретної предметної області [23].

Класифікаційні моделі, зокрема дерева рішень, метод опорних векторів (SVM), логістична регресія, а також нейронні мережі, можуть бути використані для виявлення потенційних ризиків у проєктах на основі аналізу історичних даних. Такі моделі навчаються на прикладах із попередніх проєктів, де відомо, чи виникала проблема, і які фактори цьому передували. Кожен об’єкт аналізу — наприклад, окреме завдання або етап проєкту — описується множиною ознак, що можуть включати тип завдання, його тривалість, обсяг ресурсів, кваліфікацію виконавців, ступінь залежності від зовнішніх підрядників тощо [98, 129].

Для подібних завдань часто використовується логістична регресія — одна з базових моделей класифікації, яка дозволяє оцінити ймовірність

настання певної події на основі набору вхідних ознак. Модель визначає цю ймовірність як функцію від лінійної комбінації вхідних змінних, що проходить через сигмоїдальне перетворення. Формально, ймовірність ризику можна подати у вигляді:

$$P(\text{Ризик}) = \frac{1}{1+ e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}}, \text{ де}$$

- $P(\text{Ризик})$ — ймовірність виникнення ризику.

Нейронні мережі та методи оптимізації, зокрема алгоритми генетичного програмування, демонструють високу ефективність у розв’язанні складних задач розподілу ресурсів у проєктному середовищі. Завдяки здатності до виявлення нелінійних взаємозв’язків та обробки великої кількості параметрів, ці підходи дозволяють аналізувати комбінації завдань і доступних ресурсів з метою визначення оптимальних стратегій їх розподілу [2, 78].

Алгоритми машинного навчання можуть бути ефективно застосовані для прогнозування витрат на проєкт, використовуючи історичні дані про аналогічні проєкти. Зокрема, методи регресійного аналізу або нейронні мережі здатні моделювати взаємозв’язки між різними чинниками витрат і забезпечувати точні оцінки загальних фінансових потреб. Такі моделі можуть охоплювати як прямі витрати (на робочу силу, матеріали, обладнання), так і непрямі чи непередбачувані витрати, дозволяючи здійснювати більш обґрунтоване бюджетування та знижувати ризики перевищення кошторису.

Для прогнозування тривалості проєктів і оцінки потреб у ресурсах широко використовуються методи лінійної та поліноміальної регресії, які дозволяють моделювати залежності між часовими витратами та характеристиками завдань. Класифікаційні алгоритми, зокрема дерева рішень, застосовуються для виявлення закономірностей, пов’язаних із виникненням ризиків, на основі аналізу даних з попередніх проєктів. Нейронні мережі,

завдяки своїй здатності моделювати складні, нелінійні взаємозв’язки між змінними, забезпечують більш точне прогнозування розподілу ресурсів і тривалості виконання завдань. Метод опорних векторів (SVM) часто використовується для класифікації ризиків, а також для оцінки результативності реалізованих проєктів. Алгоритми кластеризації, такі як k-середніх або ієрархічна кластеризація, дозволяють виявляти природні групи серед проєктів або завдань зі схожими характеристиками. Таке групування сприяє кращому розумінню типових сценаріїв реалізації, оптимізації ресурсів і вибору ефективних управлінських стратегій для конкретних типів проєктів [2, 24, 99].

Методи машинного навчання використовуються для аналізу історичних даних проєктів з метою виявлення закономірностей та побудови прогнозних моделей. Залежно від типу задачі, можуть застосовуватись різні підходи: регресійні моделі — для оцінки тривалості або вартості, класифікаційні — для виявлення ризиків або прогнозування успішності, а алгоритми кластеризації — для структурування даних за подібністю. У деяких випадках, наприклад, при використанні адаптивних моделей можлива автоматична адаптація моделі до нових даних. Проте якість прогнозів значною мірою залежить від повноти, релевантності та репрезентативності вхідних даних, а також від правильної постановки задачі та обраної моделі. Крім того, впровадження систем машинного навчання у процеси управління проєктами може бути досить складним і ресурсоємним. Це вимагає не лише наявності відповідної технічної інфраструктури та кваліфікованих спеціалістів, але й значних фінансових інвестицій на етапах розробки, налаштування та підтримки таких систем [25, 118].

1.4. Проблема планування проєктів

Проблема планування проєктів полягає в складності і невизначеності, які супроводжують кожен проєктний процес. При плануванні проєкту менеджерам необхідно враховувати багато факторів, таких як ресурси, терміни, бюджет, ризики, потреби стейкхолдерів та інші. Однак, в умовах постійної зміни і невизначеності, навіть найкращі плани можуть втратити актуальність або стати неефективними [33, 142].

Найбільшою проблемою є те, що багато аспектів проєкту не можуть бути точно передбачені заздалегідь. Це може включати в себе зміни вимог клієнта, технічні складнощі, зміни в бізнес-середовищі або непередбачувані події. Ці невизначеності ускладнюють процес планування та роблять навіть найкращі плани уразливими перед ризиками та змінами.

Ще однією проблемою є обмеженість ресурсів. Незалежно від того, наскільки детально розроблений план, завжди існує обмежена кількість часу, коштів та людських ресурсів, доступних для виконання проєкту. Вирішення конфліктів між ресурсами та завданнями, їх ефективне розподілення та оптимізація стають складним завданням для проєктного менеджера [6, 124].

Також, відсутність належної адаптації до змін може призвести до невдачі проєкту. Багато проєктів виконуються у динамічних умовах, де вимоги та умови змінюються з часом. Невміння адаптуватися до цих змін може призвести до затримок у графіку, перевитрат та непридатності кінцевого продукту.

Отже, проблема планування проєктів полягає в пошуку балансу між деталізацією та гнучкістю, управлінням невизначеністю та ризиками, а також ефективним використанням обмежених ресурсів. Розв'язання цих проблем потребує розробки та застосування передових методів та стратегій управління проєктами, включаючи використання алгоритмів машинного навчання та штучного інтелекту [2].

1.5. Набори даних

Серед наборів даних, які можуть використовуватися для ефективного планування проєктів, визначення пріоритетів та оцінки ризиків можна виділити наступні:

- історичні дані про попередні проєкти, що включають в себе інформацію про терміни виконання, витрати, обсяги робіт, ресурси, які використовувалися тощо. Вони дозволяють проводити аналіз та прогнозування результатів на основі досвіду минулих проєктів;
- дані про доступні ресурси, такі як людські ресурси (робочі години, навички, досвід), матеріальні ресурси (обладнання, матеріали) та фінансові ресурси. Ці дані дозволяють ефективно розподіляти ресурси для максимізації продуктивності проєкту;
- дані про окремі завдання або етапи проєкту, включаючи їх тривалість, залежності між ними, вимоги до ресурсів та інші параметри. Ці дані використовуються для розробки графіків проєкту та визначення критичних шляхів;
- дані про вимоги до продукту або проєкту, які визначають його функціональність, характеристики та очікувані результати. Вони допомагають встановити пріоритети та напрямки розвитку проєкту;
- дані про потенційні загрози та можливості, які можуть вплинути на успішність проєкту. Вони дозволяють ідентифікувати ризики та приймати заходи для їх мінімізації;
- дані про осіб або організацій, які мають інтерес або вплив на проєкт, включаючи їхні очікування, потреби та вимоги. Вони допомагають забезпечити взаєморозуміння та підтримку стейкхолдерів у процесі виконання проєкту.

Ці набори даних можуть бути зібрані з різних джерел, таких як внутрішні системи управління проєктами, зовнішні бази даних, анкети, спостереження тощо. Вони використовуються для аналізу, прогнозування та прийняття рішень у проєктному менеджменті [36, 123].

1.6. Висновки

У першому розділі було здійснено комплексний аналіз задачі високоефективного планування проєктів, що є ключовим чинником успішної реалізації проєктів у сучасному динамічному середовищі. Встановлено, що ефективне планування потребує не лише точного визначення цілей та обсягів робіт, а й врахування численних факторів — обмеженості ресурсів, динамічності зовнішнього середовища, ризиків та змін у вимогах.

У розділі здійснено огляд низки підходів до планування проєктів, зокрема методу критичного шляху (CPM), методу оцінки та аналізу програм (PERT), дерева рішень, ітеративно-інкрементальних підходів, а також методів машинного навчання.

Визначено ключову роль якісного аналізу даних у плануванні проєктів: як у частині прогнозування тривалості завдань, так і в оцінці ризиків та ефективності розподілу ресурсів. Окрему увагу приділено необхідності моніторингу ризиків у реальному часі, що передбачає використання як класичних інформаційних систем, так і інтелектуальних алгоритмів — зокрема нейромережевих моделей.

Сформульовано загальну постановку задачі як потребу в розробці нових підходів, які поєднують стратегічне планування з сучасними технологічними інструментами — зокрема, алгоритмами машинного навчання, методами прогнозування, оптимізації та прийняття рішень. Показано, що ці інструменти дозволяють обробляти великі обсяги даних, будувати прогностичні моделі та створювати адаптивні системи управління проєктами.

В свою чергу, далі запропоновано метод на основі методів машинного навчання для ефективного планування проєктів, розподілення ресурсів і завдань та оцінки ризиків. Зокрема, у наступних розділах проведено докладний розгляд, аналіз і порівняння різних підходів із запропонованим.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ПЛАНУВАННЯ ТА ОЦІНКИ РИЗИКІВ ПРОЄКТІВ

Успішна реалізація проєктів у складних і динамічних умовах вимагає не лише точного планування, але й здатності систем управління своєчасно реагувати на зміни та ризики. Незважаючи на значущість класичних методів планування, вони часто виявляються малоефективними у середовищах із високою невизначеністю.

У цьому контексті особливу увагу привертають алгоритми машинного навчання, які дозволяють будувати адаптивні моделі, здатні враховувати динаміку змін та накопичений досвід. Завдяки обробці історичних і поточних даних такі моделі створюють передумови для автоматизованого коригування планів, прогнозування ризиків і підвищення ефективності управлінських рішень.

У даному розділі запропоновано адаптивну модель управління проєктами, яка заснована на використанні машинного навчання. Модель спрямована на покращення процесу планування, а також на більш ефективне управління ризиками, що виникають під час реалізації складних проєктів.

2.1. Аналіз показників планування

Показники планування проєктів є важливим інструментом для оцінки та оптимізації виконання завдань. Одним із ключових показників є тривалість завдань, що визначає загальну тривалість проєкту. Для ефективного управління досить важливо оцінити наявні ресурси, включаючи їх доступність та ефективність використання. Пріоритетність завдань дозволяє концентрувати зусилля на особливо важливих етапах, забезпечуючи оптимальний порядок їх виконання. Використання буфера часу допомагає запобігти затримкам через непередбачувані обставини. Оцінка ефективності виконання завдань у порівнянні з початковими планами дозволяє вчасно визначати відхилення та

коригувати процес. Важливим аспектом є також збалансований розподіл навантаження між членами команди, який враховує їхню кваліфікацію, поточну завантаженість та специфіку завдань, — це дозволяє зменшити ризик перевантаження і підтримувати продуктивність на стабільному рівні.

Такий підхід дозволяє забезпечити комплексний аналіз планування і допомагає ефективно управляти як ресурсами, так і часом, що є вирішальним для успішного завершення проєкту [48, 122].

Аналіз наведених показників планування дозволяє точніше визначити оптимальні стратегії управління проєктом, але традиційні методи не завжди забезпечують необхідну гнучкість і адаптивність. Для подолання цих обмежень пропонується розробка алгоритму на базі машинного навчання, який буде враховувати зазначені показники — тривалість завдань, ресурси, пріоритетність і тип задач — та автоматично коригувати плани у відповідь на нові дані. Цей алгоритм дозволить підвищити точність прогнозування та мінімізувати ризики, пов’язані з динамікою проєкту [31].

2.2. Аналіз показників оцінки ризиків

Оцінка ризиків є важливою складовою управління проєктами, оскільки вона дозволяє ідентифікувати потенційні загрози та вчасно вжити заходів для їх мінімізації. Одним із основних показників оцінки ризиків є ймовірність виникнення події, яка може вплинути на успішність проєкту. Визначення рівня впливу таких подій допомагає оцінити, наскільки серйозно вони можуть зашкодити виконанню завдань. Інший важливий показник — це час реакції на ризик: чим швидше команда може реагувати на ризики, тим менше вони впливатимуть на проєкт. Крім того, важливо враховувати ступінь контролю над ризиком, оскільки деякі ризики є зовнішніми і не можуть бути повністю усунені [38, 116].

Для проведення глибокого аналізу ризиків необхідно використовувати як

якісні, так і кількісні методи оцінки. Це включає збір історичних даних про подібні проєкти та використання прогнозних моделей для оцінки потенційних загроз. Тут на допомогу приходять алгоритми машинного навчання, які здатні автоматично визначати взаємозв'язки між різними ризиковими факторами та оцінювати їхній вплив на проєкт. Застосування таких моделей дозволяє створювати системи, які здатні оперативно і точно оцінювати ризики та вносити корективи в плани.

Результати оцінки ризиків допомагають приймати рішення щодо розподілу ресурсів для запобігання найбільш ймовірним і значущим загрозам. Це, в свою чергу, сприяє підвищенню надійності виконання проєкту і зниженню ймовірності серйозних відхилень від плану [47, 111].

В рамках цієї роботи пропонується розробка системи, яка використовує машинне навчання для адаптивної оцінки ризиків. Завдяки алгоритмам машинного навчання, система зможе автоматично прогнозувати ризики на основі великих обсягів даних і оперативно оновлювати оцінки в реальному часі, забезпечуючи ефективне управління ризиками на всіх етапах проєкту.

2.3. Запропонований алгоритм

Запропонований алгоритм ґрунтується на багатошаровій нейронній мережі, здатній аналізувати великі масиви даних проєктного планування та оцінки ризиків. Модель побудована таким чином, що на вхідний шар подаються числові ознаки завдань — зокрема, їх тривалість, тип, пріоритет, виконавець, а також витрачений час на виконання. Ці дані нормалізуються перед обробкою для забезпечення стабільності навчання [75, 143].

Вхідні дані, що включають як історичну інформацію, так і поточний контекст виконання проєкту, використовуються для виявлення закономірностей, які можуть вплинути на часові оцінки або ймовірність

виникнення проблем під час виконання завдань [117].

Модель виконує дві основні функції:

- прогноз тривалості виконання завдання — це кількісна оцінка часу, необхідного для завершення завдання з урахуванням його характеристик, історичних патернів та ресурсного стану проєкту;
- оцінку рівня ризику — імовірнісна оцінка виникнення критичної затримки — ситуації, за якої завдання з високим або критичним пріоритетом не буде завершено в межах запланованого терміну

Ці результати інтегруються в систему планування, дозволяючи проєктним менеджерам приймати обґрунтовані рішення та швидко реагувати на зміни в умовах виконання проєкту.

2.4. Детальний опис алгоритму

Початковий етап передбачає підключення до JIRA за допомогою API для отримання даних про завдання зі спринтів. JIRA є популярним інструментом для управління проєктами, тому отримані дані відображають реальні робочі процеси команди. Після підключення через API, з JIRA можна отримати інформацію про попередні спринти. Ці дані включають імена виконавців завдань, тривалість виконання, початкові оцінки часу та пріоритети завдань тощо. Підключення до JIRA дозволяє автоматизовано отримувати дані, що спрощує процес збору інформації для прогнозів, оскільки це робиться без ручного втручання [110, 121].

Так як нейронні мережі працюють із числовими даними, категорійні дані, такі як імена виконавців і пріоритети завдань, перетворюються у числовий формат. Для цього використовуються спеціальні алгоритми перетворення, які змінюють текстові значення на числові коди. Наприклад, кожен виконавець отримує свій унікальний числовий код, що дозволяє моделі

аналізувати їх вплив на тривалість виконання завдань [76].

Також, оскільки дані про час виконання завдань можуть мати різні діапазони (одні завдання можуть тривати години, а інші — дні), необхідно нормалізувати ці значення, щоб привести їх у стандартний формат, що є зручним для навчання моделі. Цього можна досягти за допомогою масштабування, яке перетворює всі значення в діапазон [0,1]. Такий підхід дозволяє підвищити ефективність навчання моделі, оскільки вона працює з даними в одному масштабі [37].

Нейронна мережа побудована за класичною багатошаровою архітектурою: вона складається з вхідного шару, кількох щільнозв'язаних прихованих шарів, що використовують активаційну функцію ReLU, а також двох окремих вихідних нейронів. Один із них виконує задачу регресії, прогнозуючи тривалість виконання завдань, інший — задачу класифікації, оцінюючи ймовірність виникнення ризику критичної затримки [32].

Вхідний шар приймає набір характеристик завдання. Ці характеристики подаються у вигляді вектора, де кожен елемент відповідає певній характеристиці, такій як:

- виконавець (закодований як числове значення),
- пріоритет завдання,
- тип завдання (наприклад, "feature", "bug" тощо),
- початкова оцінка часу,
- витрачений час.

Припустимо, що вхідний вектор можна описати як:

$$X = [x_1, x_2, x_3, x_4 \dots x_n], \text{ де}$$

- $x_1, x_2, x_3, x_4 \dots x_n$ — це числові значення характеристик кожного об'єкта.

Модель містить два приховані шари, які використовують функцію активації Rectified Linear Unit (ReLU). Формально функція ReLU визначається як:

$$ReLu(z) = max(0, z)$$

Ця функція дозволяє уникнути проблеми «зникаючого градієнта» та сприяє ефективному навчанню глибоких нейронних мереж.

Кожен нейрон у прихованому шарі обчислює лінійну комбінацію вхідних значень:

$$z = \sum_{i=1}^{n} w_i x_i + b \text{, де}$$

- w_i — вагові коефіцієнти,
- b — зміщення (bias).

Значення z передається через функцію активації ReLU, що дозволяє моделі враховувати лише позитивні значення для подальших обчислень.

На виході мережа має два нейрони:

1. Регресійний нейрон (лінійна активація) — для прогнозування тривалості:

$$\hat{y}_{duration} = w \times z + b \text{, де}$$

- $\hat{y}_{duration}$ — прогнозоване значення тривалості.

2. Класифікаційний нейрон (сигмоїдна активація) — для оцінки ризику критичної затримки:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Значення на виході в інтервалі [0, 1] визначає ймовірність ризику затримки виконання завдання. У межах цієї моделі ризиком вважається ймовірність виникнення критичної затримки — ситуації, за якої завдання з високим або критичним пріоритетом не буде завершено в межах запланованого терміну [144].

Для навчання нейронної мережі використовуються дві різні функції втрат:

Середня абсолютна похибка (MAE) для прогнозування тривалості:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \text{ , де}$$

- N - загальна кількість спостережень у тестовій вибірці,
- y_i - фактичне значення для i-го зразка,
- $\hat{y}_i$ - прогнозоване значення для i-го зразка.
- $|y_i - \hat{y}_i|$ - абсолютна похибка прогнозу для окремого зразка [67].

Бінарна крос-ентропія для оцінки ризиків:

$$L_{risk} = -\frac{1}{N}\sum_{i=1}^{N} (y_{risk,i}\, log(\hat{y}_{risk,i}) + (1 - y_{risk,i})\, log(1 - \hat{y}_{risk,i})) \text{ , де}$$

- $y_{risk,i}$ — фактичне значення ризику для завдання i,
- $\hat{y}_{risk,i}$ — прогнозована ймовірність ризику [93].

Для оптимізації використовується алгоритм Adam (Adaptive Moment Estimation), який є розширенням методу градієнтного спуску. Він адаптивно

оновлює ваги моделі на основі оцінок першого моменту — середнього значення градієнтів та другого моменту — середнього квадрата градієнтів. На кожному кроці t градієнт функції втрат позначається як g_t , і обчислюються такі величини:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t ,$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 , \text{ де}$$

- m_t — експоненціально згладжене середнє градієнтів (перший момент),
- v_t — експоненціально згладжене середнє квадратів градієнтів (другий момент),
- β_1 , β_2 — коефіцієнти згладжування.

Щоб усунути зміщення, на початкових ітераціях використовуються скориговані оцінки:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}, \ \widehat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Оновлення ваг виконується за формулою:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \varepsilon} , \text{ де}$$

- θ_t — поточні параметри моделі,
- α — темп навчання (learning rate),
- ε — мала константа для уникнення ділення на нуль, зазвичай 10^{-8}.

Оптимізатор Adam використовується через його стійкість до зміни масштабів градієнтів і стабільність у процесі навчання. Це особливо важливо для задач прогнозування, де дані можуть бути неоднорідними за своїм масштабом, наприклад, різні оцінки часу [138].

Після навчання модель перевіряється на тестовій вибірці даних, щоб визначити, наскільки точно вона може передбачати результати. Потім нові завдання подаються на вхід моделі для прогнозування тривалості їх виконання і виникнення критичної затримки. Це дозволяє менеджерам проєктів отримувати прогнози ще до початку роботи над завданнями, допомагаючи краще планувати час і ресурси [1, 81].

2.5. Висновки

У цьому розділі було запропоновано та детально описано підхід до планування та оцінки ризиків проєктів на основі методів машинного навчання. Розглянуто ключові показники, що впливають на якість планування — тривалість завдань, ресурсне забезпечення, пріоритетність, а також метрики оцінки ризиків, зокрема ймовірність та вплив критичних затримок.

На основі аналізу сформульовано вимоги до алгоритму, який має забезпечити адаптивність до змін у проєкті та виявлення потенційних проблем до їх фактичного виникнення. Запропоновано модель глибокої нейронної мережі, яка виконує дві задачі: прогнозує тривалість виконання завдань та оцінює ймовірність виникнення критичних затримок для завдань із високим або критичним пріоритетом. Модель побудована на основі класичної багатошарової архітектури з двома вихідними нейронами для регресії та класифікації відповідно.

Алгоритм автоматично обробляє дані, отримані через API з системи JIRA, виконує масштабування вхідних ознак та трансформує категорійні значення у числовий формат. В якості функцій втрат використано середню

абсолютну похибку для задачі регресії та бінарну крос-ентропію для задачі класифікації. Для оптимізації навчання застосовується адаптивний алгоритм Adam, що забезпечує ефективне оновлення параметрів у різнорідних наборах даних.

Запропонований підхід дозволяє оперативно отримувати аналітичні прогнози щодо тривалості виконання завдань та ризиків затримок, що значно підвищує якість управлінських рішень і забезпечує проактивне управління проєктами в умовах динамічного середовища.

РОЗДІЛ 3. АРХІТЕКТУРА РОЗРОБЛЕНОЇ СИСТЕМИ

У цьому розділі детально розглядаються критично важливі етапи створення технічної інфраструктури, необхідної для розробки, навчання та тестування моделі. Презентується процес вибору та налаштування програмних компонентів, зокрема мови Python та спеціалізованих бібліотек, таких як TensorFlow, Keras та Pandas. Розглядаються методи отримання, відбору та трансформації даних, які забезпечують якісне навчання моделі та достовірність прогнозів.

У цьому розділі також представлено детальний опис розробленої системи, орієнтованої на автоматизацію процесів планування проєктів та оцінки ризиків за допомогою машинного навчання. Особливу увагу приділено компонентам для збору і обробки даних, навчання моделі та інтеграції з системами управління проєктами. Розділ завершується аналізом ключових аспектів реалізації та обговоренням висновків щодо ефективності розробленої системи. Метою цього розділу є демонстрація теоретичних і практичних рішень, що були реалізовані під час розробки системи, та надання повного уявлення про її можливості й особливості.

3.1. Налаштування середовища

Налаштування середовища для розробки моделі штучного інтелекту є ключовим етапом, що визначає продуктивність, стабільність і масштабованість усього процесу моделювання. Середовище має бути налаштоване таким чином, щоб забезпечувати ефективну роботу з великими обсягами даних і відповідати технічним вимогам навчання глибоких нейронних мереж. Основною мовою програмування обрано Python через її універсальність та розвинену екосистему спеціалізованих бібліотек, що охоплюють усі етапи життєвого циклу моделі — від попередньої обробки даних до валідації та розгортання. Для побудови нейронних мереж використовується TensorFlow,

який забезпечує створення складних і гнучких архітектур, а також підтримку апаратного прискорення за допомогою GPU. Інтерфейс Keras, як високорівневий API до TensorFlow, значно полегшує проєктування, тренування та тестування моделей. Для роботи з даними активно використовуються бібліотеки Pandas, NumPy та Scikit-learn. Такий стек інструментів забезпечує гнучке, ефективне та масштабоване середовище для розробки інтелектуальних систем [4, 5, 55].

Апаратне забезпечення відіграє важливу роль у забезпеченні ефективності та швидкості навчання моделей штучного інтелекту. Особливо критичним є використання графічних процесорів (GPU), які забезпечують значно вищу продуктивність при виконанні паралельних обчислень порівняно з центральними процесорами (CPU). Застосування GPU є практично обов’язковим при роботі з великими наборами даних і складними нейронними мережами, де обчислювальні навантаження зростають експоненційно. У випадках, коли локальне обладнання не відповідає вимогам до продуктивності, доцільним є використання хмарних рішень. Зокрема, Google Colab надає безкоштовний доступ до середовища з GPU і TPU, що дозволяє ефективно тестувати та навчати моделі з невеликим або середнім обсягом даних. Для масштабніших завдань застосовуються комерційні хмарні платформи, такі як Amazon Web Services (AWS) або Microsoft Azure, які пропонують розширені ресурси для обробки даних, масштабоване зберігання та високопродуктивні обчислювальні середовища з можливістю індивідуального налаштування [14]. Перед початком роботи з GPU перевіряється наявність драйверів CUDA та cuDNN, потрібних для роботи із TensorFlow [62, 85, 86].

Для організації ізольованого програмного середовища створюється віртуальне середовище Python. Воно гарантує стабільність і надійність роботи, а також уникає конфліктів залежностей різних проєктів. Пакети встановлюються за допомогою файлу requirements.txt, де зазначено всі

необхідні бібліотеки [5, 90]. Також можливе використання Anaconda, яка надає зручний інтерфейс для управління середовищами [15].

Для отримання актуальних даних про виконання завдань у реальних проєктах здійснюється інтеграція з системою управління проєктами JIRA. Завдяки використанню бібліотеки JIRA Python реалізується автоматизований механізм збору інформації безпосередньо з JIRA-серверів. Цей механізм дозволяє отримувати такі ключові параметри, як фактична тривалість виконання завдань, початкові оцінки, рівень пріоритету, статуси задач і призначені виконавці. Отримані дані слугують основою для формування навчальної вибірки, яка відображає особливості реальних процесів у проєктному середовищі. Такий підхід дає змогу моделі машинного навчання враховувати контекст і динаміку живих проєктів, підвищуючи точність прогнозування. Крім того, автоматизація процесу збору даних значно скорочує витрати часу, що раніше були потрібні для ручного введення або експорту даних, та забезпечує регулярне оновлення інформації, необхідної для підтримки актуальності моделі [4, 115].

Перед початком основної роботи з моделлю проводиться попереднє тестування налаштованого середовища, що є критично важливим для виявлення потенційних технічних проблем. Зокрема, здійснюється тренування простої тестової моделі з використанням TensorFlow, щоб перевірити його працездатність, а також коректну взаємодію з GPU для апаратного прискорення. Паралельно тестується стабільність підключення до JIRA API, що дозволяє гарантувати безперебійний доступ до необхідних даних у режимі реального часу. Для управління змінами та забезпечення контрольованого середовища розробки використовується система контролю версій Git [51]. Це дозволяє не лише фіксувати кожен етап модифікації коду чи конфігурації, а й забезпечує повторюваність процесу навчання, що є важливим у

науково-обґрунтованому підході до побудови моделей машинного навчання [3].

3.2. Вибір набору даних

Для успішної розробки моделі машинного навчання вирішальне значення має якість і репрезентативність навчального набору даних. У межах даного проєкту основним джерелом даних виступає система управління завданнями JIRA, яка містить докладну інформацію про хід виконання завдань у попередніх спринтах. Зазначені параметри мають ключове значення для навчання моделі, оскільки безпосередньо впливають на прогнозування тривалості завдань, оцінювання ресурсних потреб та визначення ймовірності виникнення ризиків [21, 139].

Отримання даних із системи JIRA реалізується за допомогою бібліотеки JIRA Python, яка забезпечує інтеграцію з API платформи та дозволяє автоматизувати процес збору інформації. Даний підхід забезпечує регулярне оновлення даних, що є ключовим елементом для створення адаптивної моделі [56]. Для тестування та перевірки моделі також можуть використовуватися штучно синтезовані дан а також публічно доступні набори даних з платформи Kaggle, які імітують реальні сценарії планування проєктів [63].

Для забезпечення високої якості вхідних даних перед навчанням моделі виконується ретельна попередня обробка. На цьому етапі видаляються пропущені значення, аномальні спостереження та некоректні записи, які можуть негативно вплинути на точність моделі. Далі виконується нормалізація числових параметрів, що дозволяє звести всі ознаки до єдиного масштабу. Категорійні змінні, такі як імена виконавців та типи завдань, переводяться у числовий формат для подальшої обробки нейронною мережею [10, 137].

Ретельний відбір і підготовка даних суттєво підвищують ймовірність

коректного функціонування моделі, навіть у випадках обмеженої кількості вхідної інформації.

3.3. Підготовка даних для моделі

Для забезпечення ефективного та стабільного навчання моделі всі числові змінні, зокрема початкові оцінки тривалості та фактичний час виконання завдань, проходять процедуру нормалізації. Метою цього процесу є приведення всіх значень до одного масштабу, що усуває ризик домінування змінних з великим діапазоном значень над іншими ознаками. Нормалізація виконується за наступною формулою:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \text{ де}$$

- x — вихідне значення,
- x_{max} та x_{min} — максимальне та мінімальне значення у вибірці.

Категорійні ознаки, такі як виконавці завдань, типи завдань (наприклад, "Bug", "Feature", "Improvement") та рівні пріоритетів, не можуть бути безпосередньо використані в більшості моделей машинного навчання, оскільки вони представлені у текстовому форматі. Тому їх необхідно перетворити в числовий вигляд за допомогою методів кодування категорійних змінних. Зокрема, Label Encoding надає кожній категорії унікальне числове значення, а One-Hot Encoding створює бінарні стовпці для кожної категорії. Це дозволяє моделі оптимально працювати з текстовими параметрами, перетвореними на числові значення [91, 120].

Наступним етапом підготовки даних є розподіл обробленої вибірки на навчальну, тестову та валідаційну частини. Такий розподіл дозволяє перевірити, наскільки добре модель узагальнює знання і справляється з

новими, раніше не баченими даними. Дані розподіляються у співвідношенні 70:20:10: 70% — для навчання, 20% — для тестування, 10% — для валідації. Перед розподілом виконується перемішування даних, що гарантує незалежність моделі від порядку записів у наборі.

Усі підготовлені дані перетворюються у векторну форму, де кожен вектор представляє окреме завдання з проєкту (Task Vector=[Assignee, Type, Priority, Original Estimate, Actual Time]), де кожен елемент є закодованим параметром або числовим значенням [4, 116].

3.4. Детальна структура проєкту

3.4.1. Підключення до JIRA

Цей етап передбачає інтеграцію із системою JIRA для автоматизованого отримання даних про завдання та їхні характеристики. JIRA є однією з найбільш популярних систем управління проєктами, яка забезпечує централізоване зберігання інформації про завдання, їхній статус, спринти та взаємозалежності. Інтеграція з JIRA API дозволяє отримувати необхідні дані без необхідності ручного введення, що суттєво зменшує ризики помилок і підвищує точність аналізу [96].

Процес отримання даних починається зі встановлення з’єднання із сервером JIRA за допомогою бібліотеки jira, яка надає зручний інтерфейс для взаємодії з API системи. Це дозволяє виконувати запити до системи та отримувати необхідні відомості у форматі JSON, які потім обробляються для подальшого використання у навчанні моделі.

Наступним кроком є формування запитів JQL (JIRA Query Language), що дає змогу здійснювати вибірку завдань за заданими критеріями, такими як проєкт, статус, виконавець або спринт. Наприклад, запит project = YOUR_PROJECT_KEY AND sprint in closedSprints() забезпечує отримання всіх завдань, що належать до завершених спринтів конкретного проєкту.

Завдяки цьому можна зібрати історичні дані про виконані завдання, що є основою для аналізу та навчання моделі [94].

```
from jira import JIRA

# Встановлення з'єднання з JIRA
jira_options = {'server': 'https://your-jira-instance.atlassian.net'}
jira = JIRA(options=jira_options,
            basic_auth=('your-email@example.com', 'your-api-token'))

# Формування JQL-запиту для отримання завдань із закритих спринтів
issues = jira.search_issues(
    'project = YOUR_PROJECT_KEY AND sprint in closedSprints()',
    maxResults=2500)

# Ініціалізація структури для збереження даних
data = []
```

Рис 2. Код підключення до JIRA

Після виконання запиту отримані завдання аналізуються для виділення ключових параметрів, необхідних для подальшого використання та прогнозування. До них належать:

- Assignee - виконавець завдання, що дає змогу аналізувати продуктивність конкретних фахівців та виявляти залежності між виконавцем і фактичним часом виконання.
- Task Type - категорія завдання (наприклад, "Bug", "Feature", "Improvement"), що дозволяє класифікувати завдання за їхньою природою та оцінювати вплив типу на тривалість виконання.
- Original Estimate - початкова оцінка часу виконання у годинах, що є важливим параметром для аналізу точності планування та прогнозування майбутніх оцінок.

- Time Spent - фактичний час виконання завдання у годинах, що використовується як цільова змінна для навчання моделі прогнозування тривалості.
- Priority - пріоритет завдання (наприклад, "High", "Medium"), який впливає на розподіл ресурсів та може корелювати з термінами виконання.
- Risk Flag - індикатор ризику, який використовується для позначення завдань із підвищеною ймовірністю проблем під час їх виконання. Це може бути важливим фактором при оцінці загальних ризиків проєкту.

```
# Обробка отриманих завдань

for issue in issues:
    fields = issue.fields
    data.append({
        'task_id': issue.key,
        'assignee': fields.assignee.displayName if fields.assignee else 'Unassigned',
        'task_type': fields.issuetype.name,
        'original_estimate': fields.timetracking.originalEstimateSeconds
        / 3600 if fields.timetracking.originalEstimateSeconds else 0,
        'time_spent': fields.timetracking.timeSpentSeconds
        / 3600 if fields.timetracking.timeSpentSeconds else 0,
        'priority': fields.priority.name if fields.priority else 'Medium',
        'risk_flag': compute_risk_flag(fields)
    })
```

Рис 3. Код обробки отриманих даних

У межах дослідження сформульовано умову, за якою кожному завданню зі спеціальними атрибутами призначається ознака risk_flag:

$$\text{risk\_flag} = \begin{cases} 1, & \text{якщо } (\text{priority} \in \{\text{High}, \text{Critical}\}) \ \wedge \ (\text{resolution\_date} - \text{due\_date} \geq 2 \text{ доби}) \\ 0, & \text{інакше.} \end{cases}$$

Тобто ризиковим вважається виключно те завдання, яке:

- має високий або критичний пріоритет (High або Critical);
- фактично завершується (resolution_date) пізніше від встановленої дати завершення (due_date). У даному прикладі було встановлено дельту у дві доби, що задовольняє умовам обраного набору даних, який детально описано у наступному розділі. Але це значення може бути адаптовано до умов конкретного проєкту та встановлено керівником.

Задачі з високим чи критичним пріоритетом мають пряму й вагому дію на успіх проєкту, адже їхня затримка може спричинити суттєві збої в загальному розкладі або якості продукту. Прострочення дедлайну більш ніж на дві доби свідчить про наявність реальних проблем у роботі (технічних, організаційних тощо), а отже, вважається ознакою реального ризику для виконання проєкту. Якщо завдання має нижчий пріоритет, його прострочення може бути менш критичним. Водночас, якщо задача високого чи критичного пріоритету виконується вчасно чи без значного запізнення, у ній немає сенсу вбачати високий ризик. Таким чином, одночасний контроль за пріоритетом та відхиленням від дедлайну забезпечує більш точну ідентифікацію ризикових завдань [82].

Якщо у завдання відсутні due_date чи resolution_date, тоді за умовчанням вважається risk_flag = 0. Якщо обидві дати існують, то обчислюється їхня різниця.

```python
from dateutil import parser
from datetime import timedelta

def compute_risk_flag(fields):
    """
    Обчислює risk_flag:
      1, якщо:
        - priority = High або Critical
        - resolution_date > (due_date + 2 дні)
      0, інакше.
    """
    priority_name = fields.priority.name if fields.priority else 'Medium'
    due_date_str = fields.duedate
    resolution_date_str = fields.resolutiondate

    # Якщо пріоритет не High/Critical — повернемо 0
    if priority_name not in ['High', 'Critical']:
        return 0

    # Якщо due_date або resolution_date відсутні — повернемо 0
    if not due_date_str or not resolution_date_str:
        return 0

    try:
        due_dt = parser.parse(due_date_str)
        resolution_dt = parser.parse(resolution_date_str)

        # Якщо resolution_dt перевищує due_dt більше ніж на 2 дні
        if resolution_dt > (due_dt + timedelta(days=2)):
            return 1
        else:
            return 0
    except (ValueError, TypeError) as e:
        # У разі помилки парсингу — повертаємо 0
        print(f"Помилка парсингу дат для задачі {issue.key}: {e}")
        return 0
```

Рис 4. Функція обчислення ризику

Зібрані дані формують основу для навчання моделі, оскільки вони дозволяють аналізувати історичні показники виконання завдань і прогнозувати їхню майбутню тривалість та рівень ризику. Усі отримані параметри ретельно обробляються: пропущені значення заповнюються, категорійні дані кодуються

у числовий формат, а числові змінні нормалізуються. Це забезпечує стабільність роботи нейронної мережі та покращує якість прогнозування [136].

Автоматизація цього процесу дозволяє забезпечити актуальність і точність отриманої інформації, що є критично важливим для побудови ефективної моделі прогнозування. Завдяки інтеграції з JIRA забезпечується регулярне оновлення набору даних, що дозволяє адаптувати модель до поточних умов і враховувати нові патерни у виконанні завдань.

Таблиця 1

Приклад зібраних даних

Task ID	**Assignee**	**Task Type**	**Original Estimate (h)**	**Time Spent (h)**	**Priority**	**Risk Flag**
TASK-001	John Smith	Bug	8.0	10.0	High	1
TASK-002	Jane Doe	Improvement	16.0	14.0	Medium	0
TASK-003	Unassigned	Bug	4.0	5.0	Low	0
TASK-004	Alice Ray	Feature	2.0	4.0	Critical	1

3.4.2. Попередня обробка даних

Попередня обробка даних є критично важливим етапом, який забезпечує якість і узгодженість вхідної інформації для моделі, що значно впливає на точність і стабільність її роботи. Без ретельної обробки даних нейронна мережа може працювати некоректно, оскільки нерівномірні або відсутні значення можуть спричиняти небажані відхилення у прогнозах.

Процес попередньої обробки даних охоплює кілька ключових заветапів, які спрямовані на підготовку вхідних даних до подальшого навчання моделі. Вони включають очищення даних, заповнення пропущених значень, нормалізацію числових змінних, кодування категорійних даних та формування навчального набору.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, MinMaxScaler

# Створення DataFrame із зібраних даних
data = pd.DataFrame(data)

# Очищення даних: Заповнення пропущених значень
data.fillna({'time_spent': 0, 'original_estimate': 0, 'priority': 'Medium'}, inplace=True)

# Кодування категорійних даних
le_assignee = LabelEncoder()
data['assignee_encoded'] = le_assignee.fit_transform(data['assignee'])

le_task_type = LabelEncoder()
data['task_type_encoded'] = le_task_type.fit_transform(data['task_type'])

priority_mapping = {'Low': 1, 'Medium': 2, 'High': 3, 'Critical': 4}
data['priority_encoded'] = data['priority'].map(priority_mapping)

# Нормалізація числових даних
scaler = MinMaxScaler()
data['time_spent_scaled'] = scaler.fit_transform(data[['time_spent']])
data['original_estimate_scaled'] = scaler.fit_transform(data[['original_estimate']])
```

Рис 5. Код попередньої обробки даних

Очищення даних є необхідним кроком для забезпечення коректності та узгодженості вхідної інформації. На цьому етапі виконується виявлення

пропущених значень і їхнє заповнення відповідними замінами, щоб уникнути впливу відсутніх даних на навчання моделі. Наприклад, якщо в полях time_spent (витрачений час) та original_estimate (початкова оцінка) є відсутні значення, вони автоматично видаляються, що запобігає виникненню помилок під час розрахунків. Якщо значення пріоритету завдання відсутнє, йому присвоюється середнє значення — "Medium", що дозволяє зберегти цілісність набору даних.

Далі виконується кодування категорійних змінних, щоб перетворити текстові значення у формат, придатний для обробки моделлю. Для цього використовується метод Label Encoding, за допомогою якого кожній унікальній категорії призначається числовий код. Наприклад, якщо в наборі даних містяться такі категорії як "Bug", "Feature" і "Improvement", вони кодуються як 0, 1 і 2 відповідно. Аналогічно, пріоритети завдань ("Low", "Medium", "High", "Critical") перетворюються у числові значення за допомогою спеціального словника відповідностей, де "Low" отримує значення 1, "Medium" — 2, "High" — 3, а "Critical" — 4. Це дає змогу моделі краще працювати із пріоритетними завданнями, розрізняючи їхню важливість [108].

Наступний важливий крок — нормалізація числових змінних, яка необхідна для забезпечення рівномірного впливу кожного параметра на навчання моделі. Це особливо важливо, оскільки різні змінні можуть мати значно відмінні масштаби (наприклад, одне завдання може тривати 1 годину, а інше — 100 годин). Цей підхід дозволяє знизити вплив масштабів різних змінних на процес навчання та покращує стабільність навчального процесу.

Отримані дані є підготовленими для використання в нейронній мережі, що дозволяє їй ефективно аналізувати вхідну інформацію та будувати точні прогнози.

Після всіх попередніх кроків формується результуючий набір даних, який включає в себе:

- матрицю ознак (X) — містить закодовані та нормалізовані дані про виконавця, тип завдання, початкову оцінку та пріоритет;
- цільові змінні (y) — включають інформацію про фактичний витрачений час і рівень ризику.

```
X = data[['assignee_encoded', 'task_type_encoded', 'original_estimate_scaled',
          'priority_encoded']].values
y_duration = data['time_spent_scaled'].values
y_risk = data['risk_flag'].values
```

Рис 6. Код формування результуючого набору даних

3.4.3. Формування вибірок

Розподіл даних на вибірки є важливим етапом навчання моделі, що забезпечує її коректність та узагальнювальну здатність. Основна частина даних отриманих, а саме 70%, використовується як навчальна вибірка, яка слугує базою для налаштування вагових коефіцієнтів нейронної мережі. Саме на цьому етапі модель вчиться розпізнавати закономірності та адаптувати свої параметри відповідно до вхідних даних.

Наступні 20% від загального обсягу формують тестову вибірку, яка дозволяє оцінити ефективність моделі на раніше невідомих даних. Це дає змогу перевірити, наскільки точно модель здатна прогнозувати результати в умовах, які відрізняються від навчальних.

Залишкові 10% даних відводяться на валідаційну вибірку, що виконує функцію додаткового контролю під час навчання. Вона допомагає запобігти

перенавчанню, забезпечуючи баланс між гнучкістю моделі та її здатністю узагальнювати знання для нових сценаріїв. Це гарантує, що модель не просто запам’ятовує навчальні дані, а навчається робити правильні прогнози навіть за змінних умов.

Підготовка даних забезпечує якість вхідної інформації для моделі, що є ключовим для стабільного та точного навчання.

3.4.4. Побудова нейронної мережі

Побудова нейронної мережі є ключовим етапом розробки, що визначає її здатність до точного прогнозування тривалості виконання завдань та оцінки ризиків. У межах цієї роботи створено архітектуру багатошарової нейронної мережі, що забезпечує ефективну обробку вхідних параметрів, адаптивне навчання та гнучке застосування для різних проєктних сценаріїв [70].

Основою моделі є вхідний шар, який приймає структуровані характеристики завдань, включаючи виконавця, тип завдання, початкову оцінку, пріоритет та інші фактори. Цей рівень формує вхідний вектор, що передається на подальшу обробку.

Подальша трансформація даних здійснюється в прихованих шарах, які використовують функцію активації ReLU (Rectified Linear Unit). Кількість та глибина прихованих шарів підібрані експериментально, щоб забезпечити баланс між продуктивністю та обчислювальною ефективністю [12, 135].

Таблиця 2

Архітектура моделі

Шар	**Кількість нейронів**	**Функція активації**
Вхідний шар	4	-
Прихований шар	128	ReLU
Прихований шар	64	ReLU
Прихований шар	32	ReLU
Вихідний нейрон	1	Linear
Вихідний нейрон	1	Sigmoid

Остаточний етап розробки передбачає компіляцію моделі, під час якої визначаються оптимізатор, функції втрат і метрики оцінювання. Для навчання нейронної мережі обрано оптимізатор Adam, який забезпечує швидку та ефективну мінімізацію похибок.

```
from tensorflow.keras.layers import Input
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam


inputs = Input(shape=(X.shape[1],))
x = Dense(128, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
x = Dense(32, activation='relu')(x)
output_duration = Dense(1, activation='linear', name='duration_output')(x)
output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)

model = Model(inputs=inputs, outputs=[output_duration, output_risk])
model.compile(
    optimizer=Adam(),
    loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
     metrics={'duration_output': 'mae', 'risk_output': 'accuracy'}
    )
```

Рис 7. Код побудови нейронної мережі

Розроблена архітектура забезпечує можливість одночасного розв'язання задач регресії та класифікації, що дозволяє здійснювати прогнозування тривалості виконання завдань та оцінку ймовірності виникнення критичних ризиків. З урахуванням природи вхідних даних та обмежень навчальної вибірки, результати моделі не слід інтерпретувати як абсолютно точні. Натомість, у фокусі знаходиться досягнення прийнятного рівня узагальнення та стабільно низької похибки на тестових і валідаційних підвибірках.

3.4.5. Навчання моделі на історичних даних

Навчання нейронної мережі здійснюється на основі історичних даних, зібраних із JIRA проєктів компанії Codillas GmbH. Розмір, особливості та деталі набору даних описуються у наступноу розділі.

```
# Навчання моделі
history = model.fit(
    X_train,
    {'duration_output': y_train_duration, 'risk_output': y_train_risk},
    validation_data=(X_val,
        {'duration_output': y_val_duration,
         'risk_output': y_val_risk}),
    epochs=10,
    batch_size=32,
    verbose=1
)
```

Рис 8. Код навчання моделі

Під час навчання моделі визначаються ключові параметри, що впливають на ефективність її оптимізації та точність прогнозування.

Зокрема, експерементальним чином встановлюється кількість ітерацій (epochs), протягом яких модель повністю проходить через увесь навчальний набір даних. Обране значення забезпечує достатню кількість оновлень вагових коефіцієнтів без ризику перенавчання [105].

Критичну роль відіграє також кількість вхідних ознак, що визначає набір параметрів, які модель використовує для аналізу. У цій нейронній мережі розглядається чотири основні характеристики завдань, зокрема: закодований ідентифікатор виконавця, категорія завдання, нормалізована початкова оцінка часу виконання, пріоритетність завдання [128].

Для контролю прогресу навчання відображаються метрики: середня абсолютна похибка (MAE) та точність (Accuracy).

Метрика MAE вимірює середню величину відхилення прогнозованих значень від фактичних. На відміну від інших метрик, таких як середньоквадратична похибка (MSE), MAE не підсилює вплив великих похибок, оскільки не зводить до квадрату різницю між прогнозом і реальним значенням. Таким чином, вона надає більш інтерпретовані результати, особливо якщо є аномальні значення [27, 134].

3.6. Оцінка моделі та прогнозування на нових даних

Після завершення процесу навчання модель перевіряється на тестовій вибірці, щоб оцінити її здатність до узагальнення. Окрім цього, модель використовується для прогнозування тривалості та оцінки ризиків на нових даних, що дозволяє перевірити її практичну ефективність у реальних сценаріях.

```
# Нові дані для прогнозування
new_task = pd.DataFrame({
    'assignee': ['Alice Ray'],
    'task_type': ['Feature'],
    'original_estimate': [5],  # Години
    'priority': ['High']
})

# Обробка нових даних
new_task['assignee_encoded'] = le_assignee.transform(new_task['assignee'])
new_task['task_type_encoded'] = le_task_type.transform(new_task['task_type'])
new_task['priority_encoded'] = new_task['priority'].map(priority_mapping)
new_task['original_estimate_scaled'] = scaler.transform(new_task[['original_estimate']])

# Формування вектора ознак
X_new = new_task[['assignee_encoded', 'task_type_encoded',
                  'original_estimate_scaled', 'priority_encoded']].values

# Прогнозування
new_predictions = model.predict(X_new)
predicted_duration_scaled = new_predictions[0].flatten()[0]

# Денормалізація
predicted_duration = scaler.inverse_transform([[predicted_duration_scaled]])[0][0]
# Отримуємо ймовірність ризику
predicted_risk_prob = new_predictions[1].flatten()[0]

# Вивід
print(f"Прогнозована тривалість: {predicted_duration:.2f} год")
print(f"Ймовірність критичної затримки: {predicted_risk_prob:.2%}")
```

Рис 9. Код оцінки продуктивності на нових даних

Припустимо, що в системі управління проєктами з'являється нове завдання, для якого необхідно спрогнозувати очікуваний час виконання та рівень ризику. Вхідні параметри завдання включають такі характеристики:

- виконавець: Alice Ray,

- тип завдання: Новий функціонал (Feature),
- початкова оцінка часу: 5 годин
- пріоритет: Високий

На основі цих даних модель нейронної мережі аналізує попередній досвід виконання подібних завдань, враховує історичні дані про продуктивність виконавця, середні відхилення від початкових оцінок та потенційні ризики. Модель зможе надати вихідні дані:

- очікувана тривалість виконання: 5.69 години;
- ризику критичної затримки: 5.72%

Прогнозована тривалість становить 5.69 години, що трохи перевищує початкову оцінку в 5 годин на 14%. Це може свідчити про те, що на основі історичних даних модель врахувала деякі неточності початкової оцінки, можливо через тип задачі або досвід виконавця.

Рівень ризику критичної затримки становить лише 5.72%. Це вказує на те, що, незважаючи на незначне перевищення оцінки часу, завдання має мінімальний ризик серйозного відхилення від запланованих термінів.

3.7. Архітектура проєктної системи

Архітектура розробленої системи управління проєктами побудована за принципами масштабованості, модульності та інтегрованості, що забезпечує її ефективність у динамічних умовах. У її основі лежить інтелектуальний оркестратор проєктів, здатний автоматично прогнозувати тривалість виконання завдань, оцінювати потенційні ризики та оптимізувати розподіл ресурсів за допомогою машинного навчання [7].

Для опису архітектури використано підхід C4-моделі, яка поділяє систему на чотири рівні: контекстний рівень (L1), рівень контейнерів (L2), рівень компонентів (L3) та рівень взаємодії класів (L4). Такий підхід дозволяє

структуровано пояснити, як працює система на різних рівнях деталізації. У роботі буде розглянуто перші два рівні [26].

На контекстному рівні система визначається у її загальному середовищі та взаємодії із зовнішніми учасниками. Основним користувачем є менеджер проєкту, який використовує систему для прогнозування термінів виконання завдань та оцінки можливих ризиків. Крім того, розробники взаємодіють із системою для отримання інформації про майбутні спринти та внесення змін до статусу завдань [125].

Рис 10. Контекстна діаграма

На основі історичних даних про виконані завдання даних модуль AI Risk & Duration Estimation аналізує попередні результати виконання завдань і прогнозує їхню тривалість у майбутніх проєктах.

На наступному рівні деталізації архітектуру можна розглянути з точки зору контейнерів, які відповідають за виконання основних функцій системи.

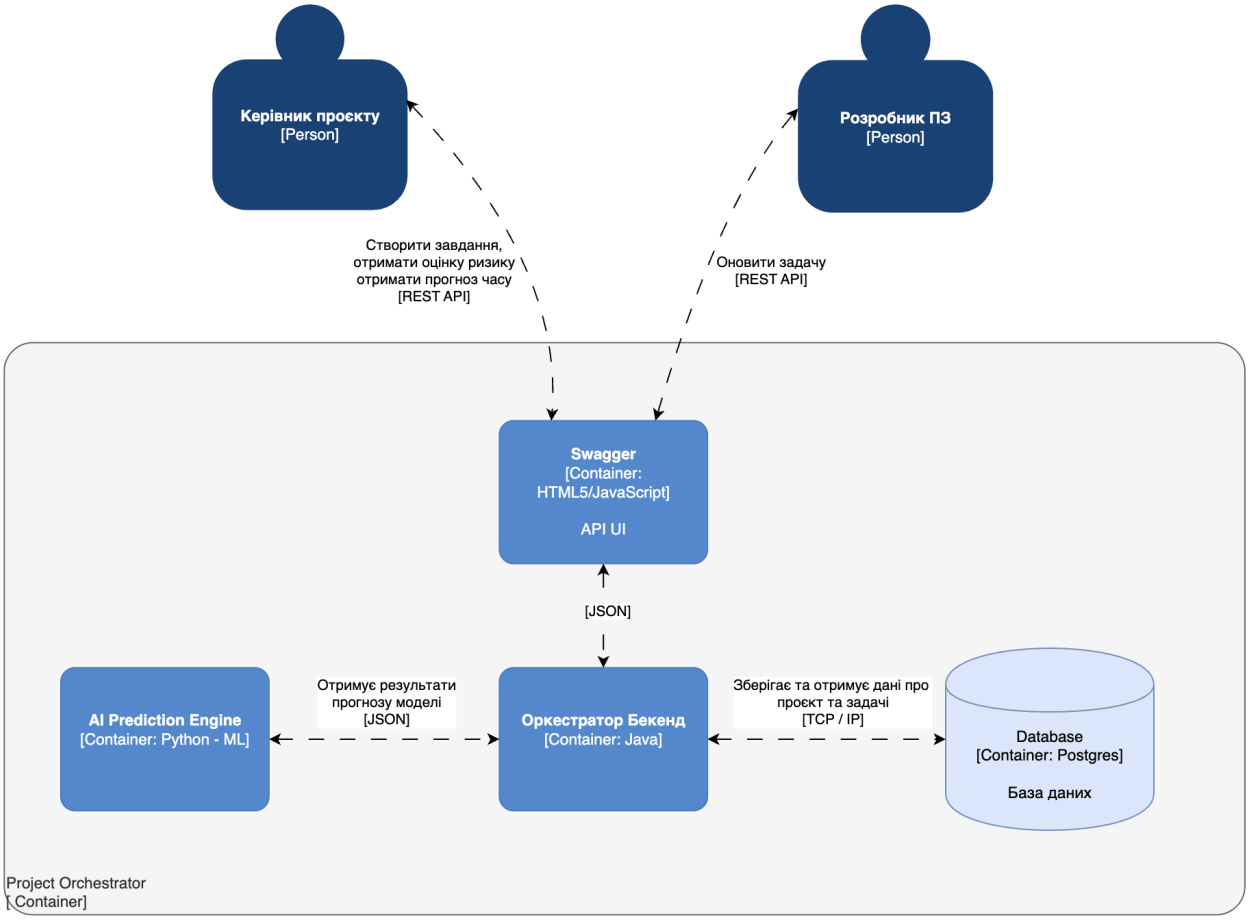

Рис 11. Діаграма контейнерів

3.7.1. Swagger

Комунікація між користувачем і системою відбувається через HTTP-запити. Це дозволяє здійснювати виклики до API з різних клієнтів, таких як веб-інтерфейси, мобільні додатки або сторонні сервіси [44].

Взаємодія з API побудована на основі RESTful-підходу, що забезпечує стандартизований обмін даними за допомогою методів HTTP:

- GET – для отримання інформації про завдання, прогнозовані терміни та ризики;
- POST – для створення нових записів, зокрема додавання нового плану спринту;
- PUT – для оновлення інформації, наприклад зміни статусу завдань;

- DELETE – для видалення записів.

Swagger є частиною OpenAPI Specification (OAS), яка визначає стандартизований підхід до опису REST API. Основна функція Swagger API – це автоматичне документування сервісів, що дозволяє розробникам легко тестувати та інтегрувати API без потреби в ручному написанні документації. У розробленій архітектурі Swagger API документує всі доступні HTTP-endpoints (кінцеві точки доступу) разом із їхніми параметрами, форматами запитів та відповідей, а також кодами статусу [88].

Swagger UI надає візуальний інтерфейс, де можна безпосередньо відправляти HTTP-запити та перевіряти їхню роботу без необхідності використовувати сторонні клієнти, такі як Postman або curl.

3.7.2. Оркестратор Бекенд (Orchestrator Backend)

Orchestrator Backend виступає як центральний серверний компонент, який забезпечує керування всіма основними процесами в системі, включаючи обробку запитів користувачів, управління базою даних, інтеграцію з модулями штучного інтелекту та координацію API. Це головний логічний шар архітектури, який відповідає за коректну взаємодію між клієнтськими запитами, модулями аналізу даних та зовнішніми сервісами [49].

Orchestrator Backend реалізовано на основі Spring Boot – потужного фреймворку для розробки високопродуктивних мікросервісних додатків мовою Java (версія 21+). Spring Boot забезпечує модульну структуру коду, легку інтеграцію з базами даних, автоматичне конфігурування, а також підтримку REST API через Spring Web [109, 119].

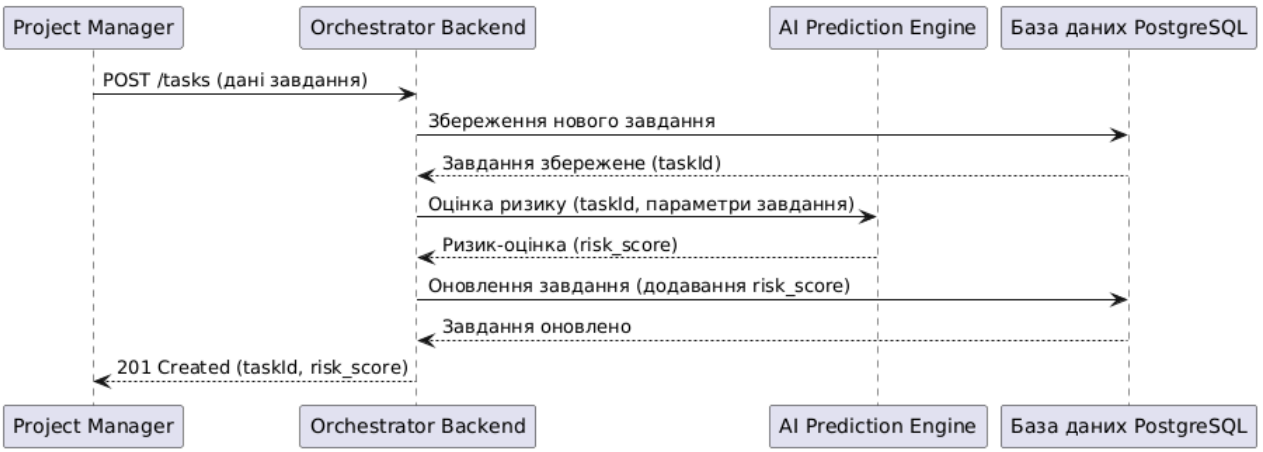

Рис 12. Діаграма послідовності: створення нового завдання

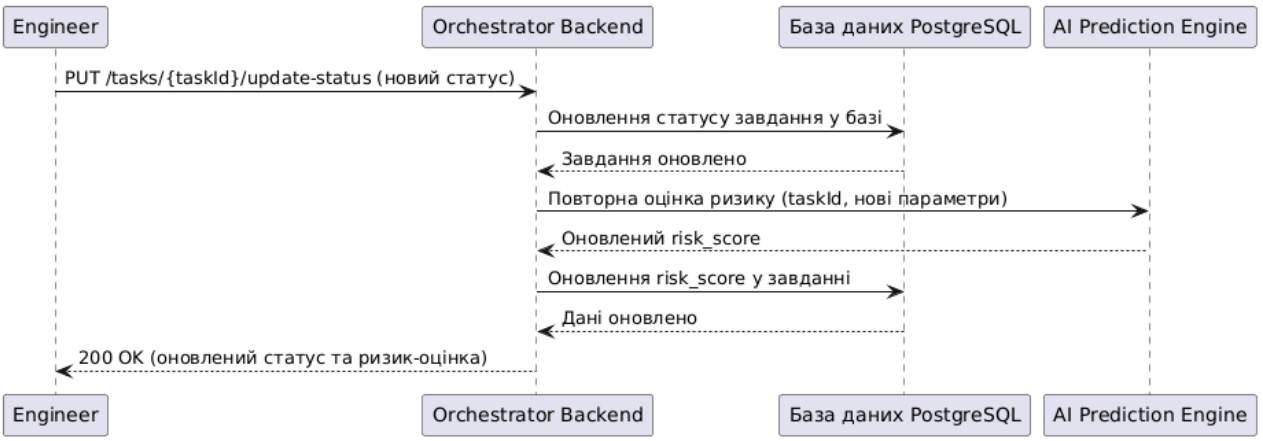

Рис 13. Діаграма послідовності: оновлення статусу завдання

Orchestrator Backend є посередником між клієнтськими запитами та модулем AI Prediction Engine, який виконує аналіз ризиків і прогнозує тривалість виконання завдань. Це забезпечує розділення логіки, що дозволяє незалежно оновлювати модель машинного навчання без впливу на основний API.

Для захисту API та забезпечення авторизованого доступу використовується Spring Security з підтримкою JWT (JSON Web Token) [61]. Це гарантує безпечну взаємодію клієнтів із сервером, дозволяючи лише

автентифікованим користувачам отримувати доступ до прогнозів та управління завданнями.

3.7.3. База даних

PostgreSQL — це потужна, відкрита система керування базами даних (RDBMS), яка використовується для зберігання та управління даними в системі AI Project Orchestrator. Вона забезпечує високу продуктивність, надійність та підтримку складних запитів, що є критично важливим для збереження історичних показників, прогнозованих ризиків та деталей завдань.

Для пришвидшення виконання складних запитів PostgreSQL пропонує розвинену систему індексації, що включає B-Tree, GIN, GiST та BRIN. Це дає змогу виконувати запити на пошук історичних показників, прогнозів ризиків та оцінок тривалості завдань із мінімальними витратами ресурсів. Наприклад, використання GIN-індексів значно прискорює пошук у JSONB-структурах, а BRIN-індекси дозволяють ефективно працювати з великими обсягами часових даних.

Ще однією важливою характеристикою PostgreSQL є підтримка паралельного виконання запитів, що суттєво підвищує продуктивність при роботі з великими обсягами інформації. Це особливо важливо для аналізу історичних даних про виконання завдань та прогнозування ризиків у режимі реального часу. Завдяки цьому запити можуть виконуватися швидше, що зменшує навантаження на сервер і покращує загальну продуктивність системи.

Окрім того, масштабованість PostgreSQL дозволяє адаптувати базу даних до зростаючих потреб проєкту. Підтримка механізмів реплікації та можливість горизонтального масштабування гарантують стабільну роботу навіть у випадку збільшення обсягу даних чи зростання кількості користувачів. Це робить PostgreSQL не лише надійним сховищем даних, а й ефективним

інструментом для довгострокового використання в системах прогнозування та управління проєктами [92].

3.7.4. AI Prediction Engine

AI Prediction Engine є ключовим компонентом AI Project Orchestrator, який відповідає за інтелектуальний аналіз вхідних параметрів і прогнозування критично важливих показників, таких як оцінка ризиків і прогнозування тривалості виконання завдань. Використовуючи методи машинного навчання та глибокі нейронні мережі, цей модуль забезпечує адаптивність і здатність до самонавчання, що дозволяє йому покращувати точність прогнозів із часом [83, 133].

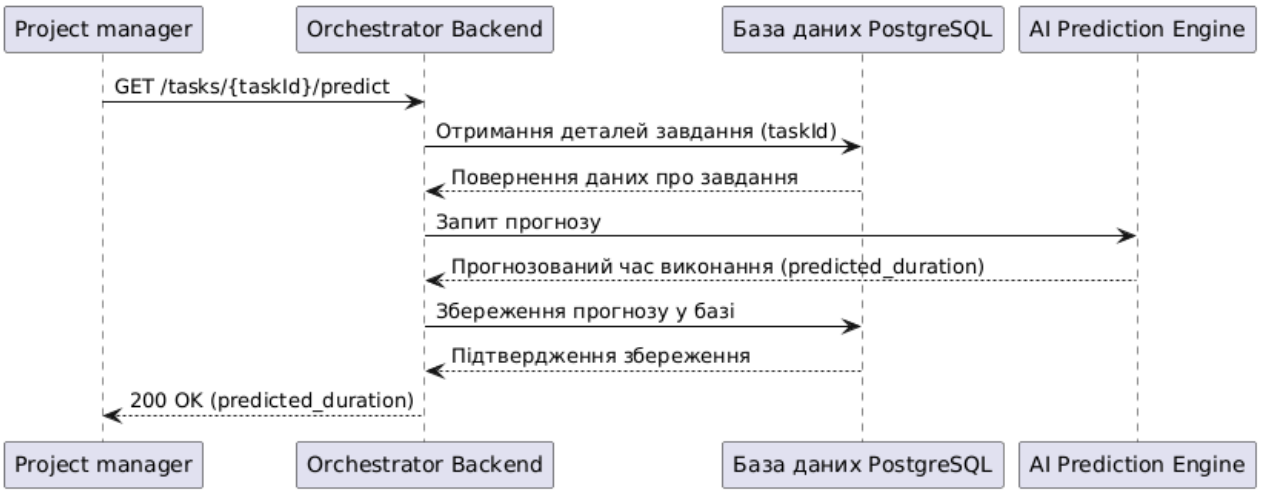

Рис 14. Діаграма послідовності: запиту прогнозу

Основне завдання цього модуля полягає в обробці структурованих та неструктурованих даних для виявлення прихованих закономірностей та кореляцій між параметрами завдань. Система використовує історичні дані про виконані завдання, щоб створити предиктивні моделі, здатні адаптуватися до поточних змін у проєктах.

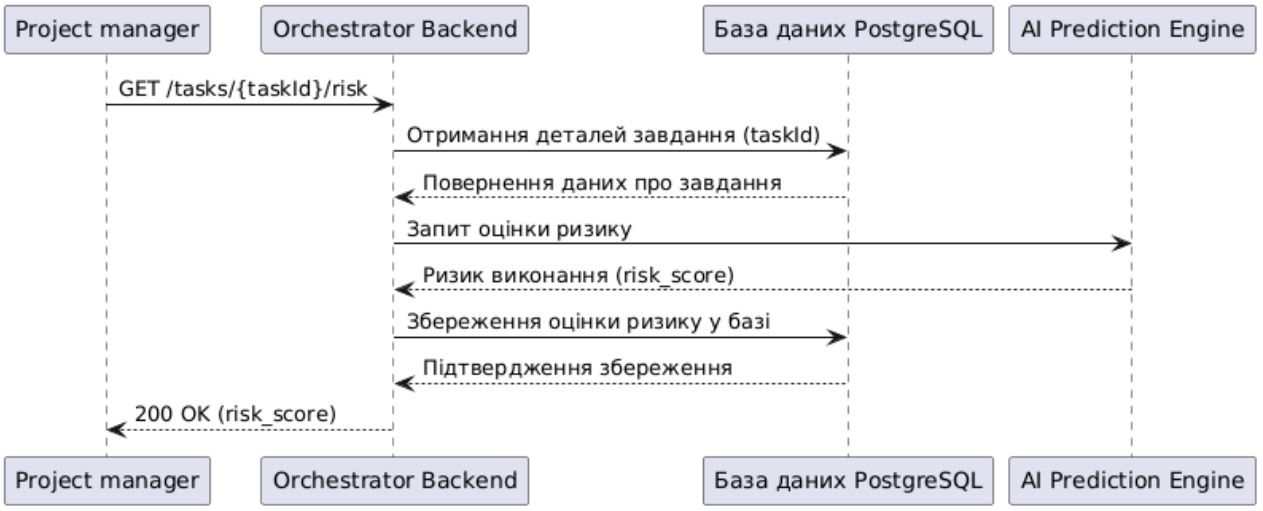

Рис 15. Діаграма послідовності: запиту ризику

AI Prediction Engine взаємодіє з іншими модулями через REST API, що дозволяє отримувати вхідні параметри у форматі JSON, обробляти їх за допомогою нейронної мережі та повертати результати прогнозування у вигляді HTTP-відповіді.

3.8. Підключення та інтеграція

API AI Project Orchestrator є основним засобом інтеграції з іншими системами, дозволяючи менеджерам проєктів, командам розробників та аналітичним інструментам автоматизовано взаємодіяти з модулями прогнозування, оцінки ризиків та управління завданнями. API побудовано відповідно до специфікації OpenAPI 3.0, що забезпечує його стандартизованість, простоту документування та підтримку широкого спектра клієнтських додатків.

API реалізовано відповідно до RESTful-архітектури, що забезпечує гнучкість у використанні, простоту інтеграції з існуючими системами та високу масштабованість [107]. Кожен ендпоінт відповідає за конкретну операцію із завданнями, що дозволяє легко інтегрувати API у корпоративні

системи, такі як JIRA, Trello, Asana або GitHub Projects [29].

API проєкту було описано за стандартами OpenAPI у форматі yaml [22].

```yaml
openapi: 3.0.0
info:
  title: AI Project Orchestrator - API
  version: 1.2.0
  description: API для управління завданнями, прогнозування часу виконання та
      оцінки ризиків.

tags:
  - name: Prediction and Risk Assessment
    description: Оцінка ризиків
  - name: Tasks
    description: Управління завданнями

paths:

  /tasks/{taskId}/risk:
    get:
      summary: Оцінка ризику виконання завдання
      description: Використання AI Prediction Engine для оцінки ймовірності
          ризику.
      tags:
        - Prediction and Risk Assessment
      parameters:
        - in: path
          name: taskId
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Оцінка ризику успішно виконана
          content:
            application/json:
              schema:
                type: object
                properties:
                  risk_score:
                    type: number
                    description: Ймовірність ризику виконання завдання (0-1)
        '404':
          description: Завдання не знайдено
```

Рис 16. Частина API схеми проєкту

API надає набір інтерфейсів, що дозволяють:

- керувати життєвим циклом завдань, створюючи, оновлюючи та отримуючи інформацію про них;
- прогнозувати тривалість виконання завдань на основі історичних даних та моделей машинного навчання;
- оцінювати ризики виконання завдань у режимі реального часу.

Компоненти API включають такі сутності:

- Tasks - управління завданнями;
- Prediction and Risk Assessment - оцінка ризиків.

3.8.1. Управління завданнями

Завдання (Tasks) є основними одиницями роботи у проєкті. API надає можливість змінювати статуси завдань, що дозволяє моделі прогнозування оновлювати оцінки ризиків у режимі реального часу.

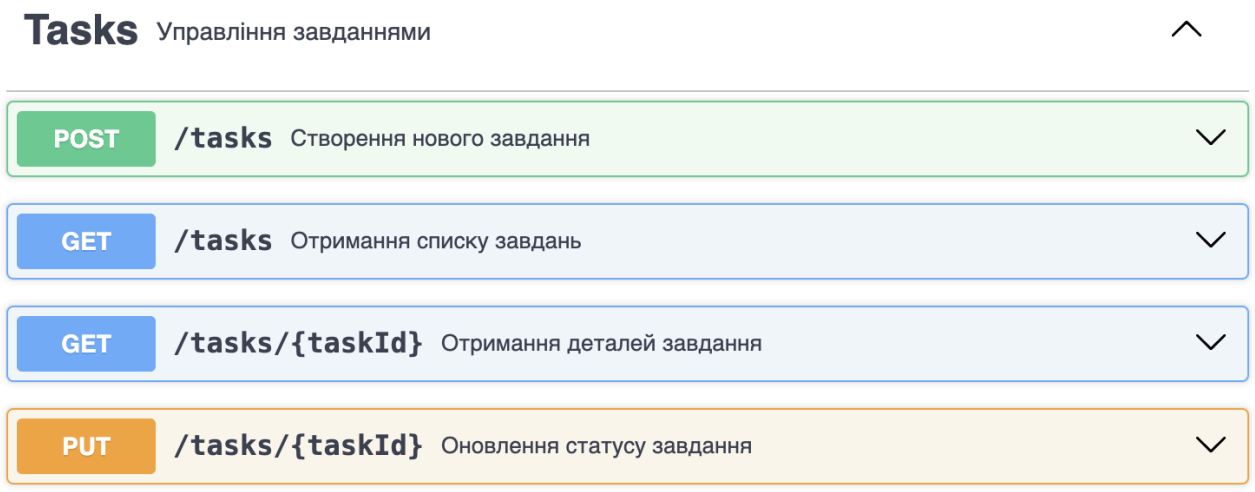

Рис 17. Swagger UI - Ендпоінти для керування завданнями

Запит POST /tasks дозволяє додати нове завдання в систему, вказавши його характеристики, включаючи назву, опис, початкову оцінку, виконавця та

статус. Всі завдання мають унікальний ідентифікатор (UUID), що забезпечує їхню ідентифікацію [71].

Запит GET /tasks повертає перелік усіх активних завдань, дозволяючи менеджерам отримувати актуальну інформацію про їхній стан.

API дозволяє отримати деталі про конкретне завдання, використовуючи GET /tasks/{taskId}. Запит повертає всі доступні характеристики завдання, включаючи його поточний статус, оцінку тривалості, пріоритет та оцінку ризику.

Запит PUT /tasks/{taskId} дозволяє оновити статус завдання, наприклад, змінити його з "in progress" на "done". Це дає змогу команді вчасно відстежувати етапи виконання та отримувати оновлену інформацію.

3.8.2. Прогнозування та оцінка ризиків

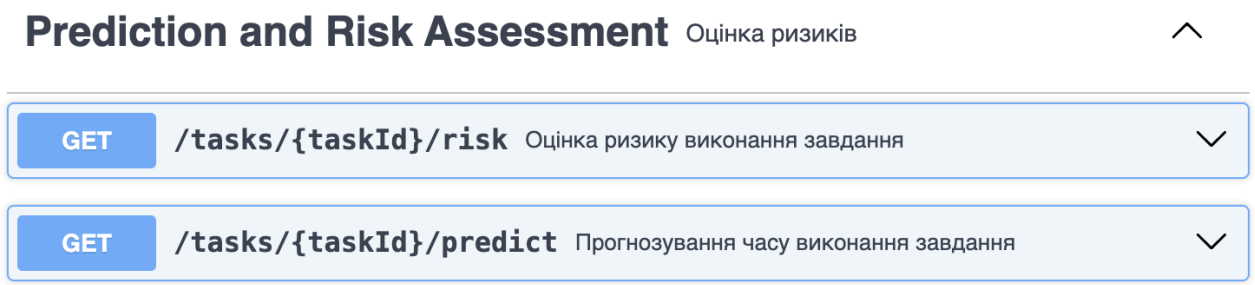

Рис 18. Swagger UI - Ендпоінти для прогнозування тривалості та оцінки ризиків

Запит GET /tasks/{taskId}/predict передає інформацію до AI Prediction Engine. Після отримання параметрів завдання, AI Prediction Engine проводить аналіз, використовуючи навчені моделі машинного навчання, що дозволяє отримати прогнозований час виконання. Отримані дані повертаються у вигляді числового значення, що вказує на передбачувану тривалість виконання в годинах.

Оцінка ймовірності ризику затримки виконання завдання здійснюється

через GET /tasks/{taskId}/risk. Цей запит дозволяє отримати прогноз щодо потенційних проблем, які можуть виникнути в процесі виконання завдання. Модель генерує ризиковий бал, що відображає ймовірність виникнення проблем у виконанні. Повернене значення варіюється в діапазоні від 0 до 1, де 0 означає низький рівень ризику, а 1 – високу ймовірність затримки виконання завдання.

3.9. Висновки

Даний розділ було присвячено опису процесу налаштування середовища, підготовці даних дня навчання моделі, а також представлено детальний опис розробленої системи.

Використання Python як основної мови програмування, разом із бібліотеками TensorFlow, Keras, Pandas, NumPy і Scikit-learn, забезпечує ефективну реалізацію нейронної мережі та обробку великих обсягів даних. Крім того, особливу увагу приділено апаратному забезпеченню: використання графічних процесорів (GPU) значно прискорює процес навчання, а хмарні сервіси (Google Colab, AWS, Azure) дають змогу масштабувати обчислювальні ресурси. Правильне налаштування віртуального середовища та інтеграція з JIRA для збору реальних даних створюють стабільну основу для подальших етапів розробки.

Розроблена система управління завданнями та оцінки ризиків спирається на сучасні технології машинного навчання та мікросервісну архітектуру, що забезпечує її масштабованість, продуктивність і надійність. У цьому розділі було детально розглянуто її архітектуру, включаючи інтеграцію API, взаємодію між компонентами та використання нейронної мережі для оцінки ризиків критичної затримки та прогнозування тривалості виконання завдань.

Архітектура цієї системи розроблена з урахуванням гнучкості та масштабованості, що дозволяє їй обробляти великі обсяги даних у

високонавантажених середовищах. Використання мікросервісного підходу забезпечує ізольованість компонентів та можливість додавання нових модулів без зміни основної логіки роботи системи.

Ключовими елементами архітектури є Orchestrator Backend, AI Prediction Engine, база даних PostgreSQL та API-інтерфейс Swagger, що забезпечують ефективну взаємодію між модулями. Оркестратор виконує роль основного комунікаційного вузла, отримуючи запити від користувачів та передаючи їх до відповідних сервісів. AI Prediction Engine реалізує алгоритми машинного навчання, обробляючи історичні дані про виконані завдання для прогнозування часу виконання та оцінки ризиків. PostgreSQL забезпечує збереження структурованої інформації, включаючи історію змін завдань, прогнозовані показники та інші дані, необхідні для навчання моделі.

Таким чином, запропонована архітектура поєднує в собі гнучкість, продуктивність та надійність, що робить її ефективним рішенням для автоматизації планування проєктів та управління ризиками. Використання сучасних технологій машинного навчання, контейнеризації та мікросервісного підходу гарантує стабільну роботу системи навіть при значному навантаженні та зростанні кількості користувачів.

РОЗДІЛ 4. ЗАСТОСУВАННЯ МЕТОДУ ТА ПЕРЕВІРКА ЯКОСТІ

Запропонована система планування завдань і оцінки ризиків на основі методів машинного навчання була реалізована та протестована на реальних наборах даних наданих компанією Codillas GmbH. У цьому розділі детально розглянуто якість запропонованого методу, включаючи опис використаних даних, аналіз ефективності прогнозування, порівняння з іншими підходами та оцінку доцільності впровадження цієї методики в управлінні проєктами. Основна мета цього розділу — довести ефективність розробленого підходу та оцінити його переваги над традиційними методами управління проєктами.

4.1. Опис набору даних

У межах дисертаційного дослідження було використано реальний набір даних, наданий компанією Codillas, яка спеціалізується на розробці програмного забезпечення та впровадженні цифрових рішень. Датасет містить історичні дані про виконання задач, які формуються у внутрішній системі керування проєктами (JIRA). Він охоплює задачі різного типу, що були створені, оцінені та завершені протягом звітного періоду в рамках декількох проєктних команд.

Для забезпечення надійності аналітики було проведено ретельну попередню обробку набору даних.

task_id	assignee	task_type	original_estimate	time_spent	priority	due_date	resolution_date
TASK-1286	Diana Prince	Bug	34	36	Medium	2023-07-10	2023-07-10
TASK-1287	Jane Doe	Improvement	20	21	Medium	2023-02-20	2023-03-07
TASK-1288	Alexander Grey	Research	30	28	Low	2023-02-26	2023-02-26
TASK-1289	John Doe	Refactor	36	33	Low	2023-07-07	2023-07-06
TASK-1290	John Doe	Bug	39	35	High	2023-06-19	2023-06-26
TASK-1291	Charlie Brown	Research	17	18	Low	2023-06-13	2023-06-13
TASK-1292	Diana Prince	Research	2	2	Medium	2023-05-21	2023-05-20
TASK-1293	Alexander Grey	Refactor	13	14	High	2023-04-18	2023-04-21
TASK-1294	George Cole	Improvement	21	19	Critical	2023-07-10	2023-07-05
TASK-1295	Bob Smith	Research	25	28	Medium	2023-07-14	2023-07-09
TASK-1296	Alice Ray	Refactor	37	36	High	2023-02-26	2023-02-22
TASK-1297	Alice Ray	Improvement	8	7	Critical	2023-02-09	2023-02-05
TASK-1298	Fiona Miller	Feature	4	4	Low	2023-01-01	2023-01-07
TASK-1299	Fiona Miller	Bug	17	18	High	2023-03-28	2023-04-02
TASK-1300	Alexander Grey	Bug	18	18	Medium	2023-02-07	2023-02-06
TASK-1301	George Cole	Improvement	4	3	Low	2023-01-03	2023-01-04
TASK-1302	Evan Stone	Refactor	33	33	Medium	2023-04-14	2023-04-14
TASK-1303	Jane Doe	Improvement	30	32	Critical	2023-07-09	2023-07-04
TASK-1304	John Doe	Research	36	38	High	2023-01-25	2023-01-22
TASK-1305	Charlie Brown	Research	29	26	Low	2023-05-26	2023-05-29
TASK-1306	Alexander Grey	Refactor	1	1	Low	2023-06-02	2023-05-30
TASK-1307	Charlie Brown	Feature	1	1	High	2023-01-03	2022-12-31
TASK-1308	Charlie Brown	Research	28	28	Critical	2023-05-15	2023-05-10
TASK-1309	Bob Smith	Bug	30	33	Medium	2023-03-09	2023-03-04

Рис 19. Частина датасету від компанії Codillas GmbH

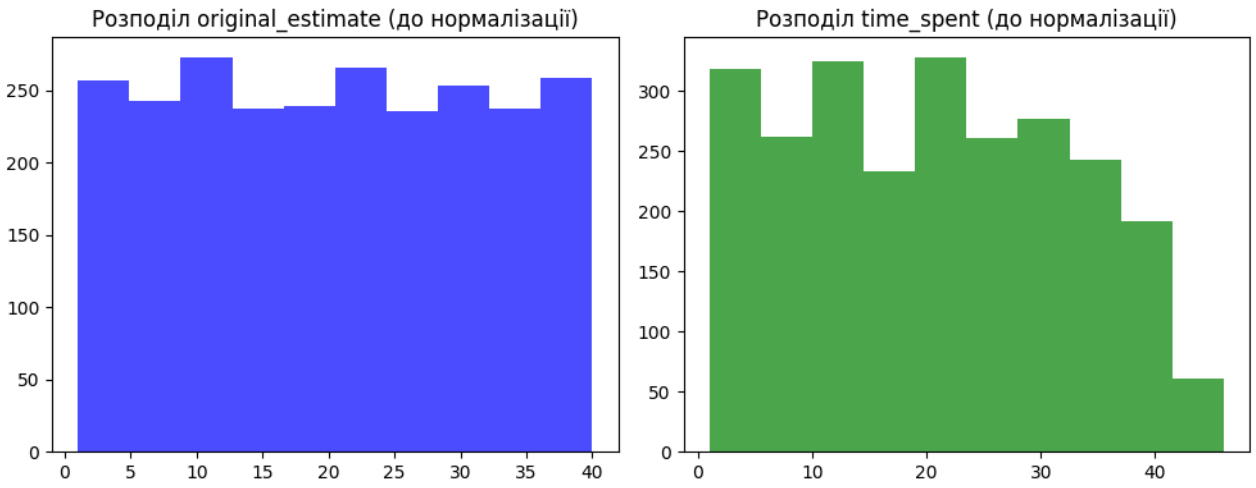

Рис 20. Розподіл даних до нормалізації

Для підвищення аналітичної придатності даних було реалізовано низку процедур попередньої очистки. Насамперед, було видалено усі записи з очевидними аномаліями, зокрема, такі, де витрачений час дорівнював нулю при наявності завершення задачі, або випадки, коли фактична тривалість

перевищувала початкову оцінку більш ніж утричі. Також було усунено записи з хибними або непридатними до парсингу датами, а також здійснено нормалізацію форматів усіх текстових та числових полів. Таким чином, сформовано очищену та структуровану вибірку для подальшого моделювання [74].

Аналіз розподілу значень у масиві демонструє, що середнє значення планової тривалості (original_estimate) склало 20.9 години, що вказує на те, що більшість задач належать до категорії середньої складності. Середній фактично витрачений час (time_spent) становить 21.0 годин, що свідчить про загалом адекватне планування та помірний рівень відхилення у часі. Проте у 44.3% задач зафіксовано перевищення фактичного часу над запланованим, що вказує на необхідність ретельнішого аналізу впливу різних факторів на точність оцінки.

Особливу увагу в дослідженні приділено аналізу задач, виконання яких супроводжувалося суттєвим ризиком. Ризиковими вважаються задачі, які одночасно мають високий або критичний рівень пріоритетності та завершені з перевищенням дедлайну більше ніж на два календарні дні. Частка таких задач у досліджуваному масиві склала 15.7%, що є достатньою для статистично релевантного моделювання. При цьому, було зафіксовано 33.2% випадків перевищення терміну виконання, незалежно від рівня пріоритету, що свідчить про наявність загальних системних факторів впливу на дотримання строків.

Також було виявлено наступну закономірність: задачі типу Research, Refactor та Bug, особливо з низьким або середнім рівнем пріоритетності, мають вищу імовірність виникнення затримок. Це може бути пов’язано з тим, що такі задачі зазвичай відкладаються на користь більш пріоритетних або бізнес-критичних завдань, що призводить до накопичення технічного боргу або затримок у дослідженнях. Крім того, аналіз індивідуальних виконавців показав, що деякі співробітники — зокрема, Bob Smith, Evan Stone та Diana

Prince — мають вищу імовірність затримки виконання задач високого рівня пріоритетності, що дозволяє враховувати фактори персональної ефективності в контексті прогнозування ризиків.

Отриманий набір даних є репрезентативним, збалансованим за ключовими параметрами та достатньо варіативним для побудови моделей машинного навчання з метою прогнозування часових характеристик задач та оцінки їх ризиковості.

Таблиця 3

Вигляд оброблених даних

Assignee Encoded	Task Type Encoded	Original Estimate Scaled	Time Spent Scaled	Priority Encoded	Risk Flag
0	1	0.25	0.30	3	1
1	2	0.75	0.65	2	0
2	0	0.50	0.45	4	1
3	3	0.25	0.50	5	1

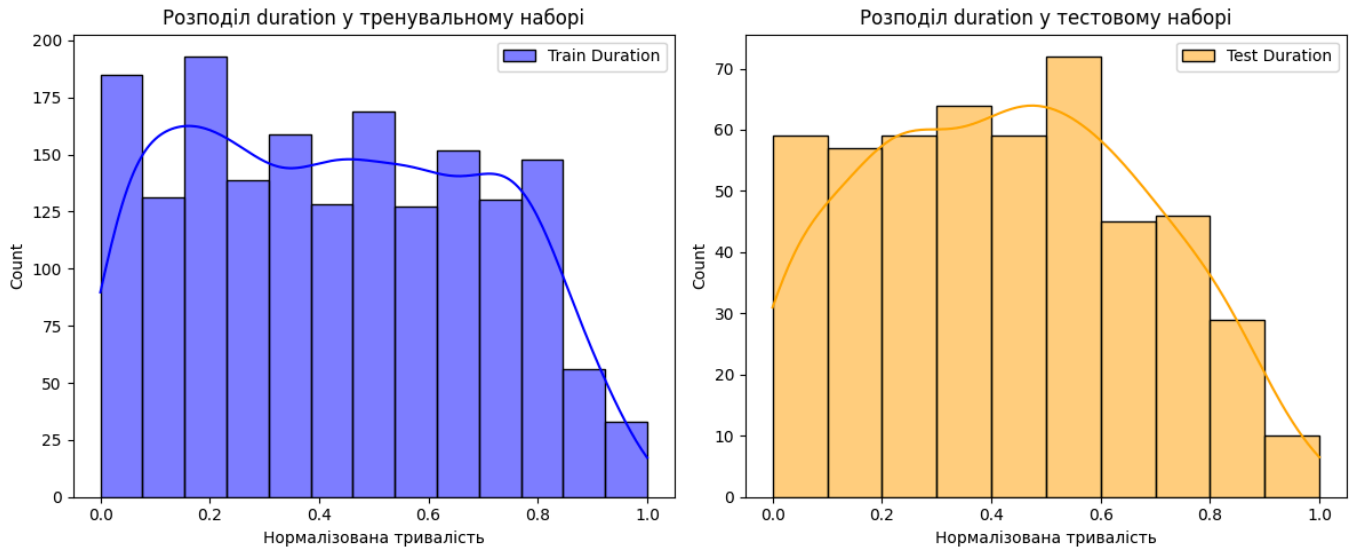

Рис 21. Розподіл нормалізованої тривалості

у тренувальному та тестовому наборах

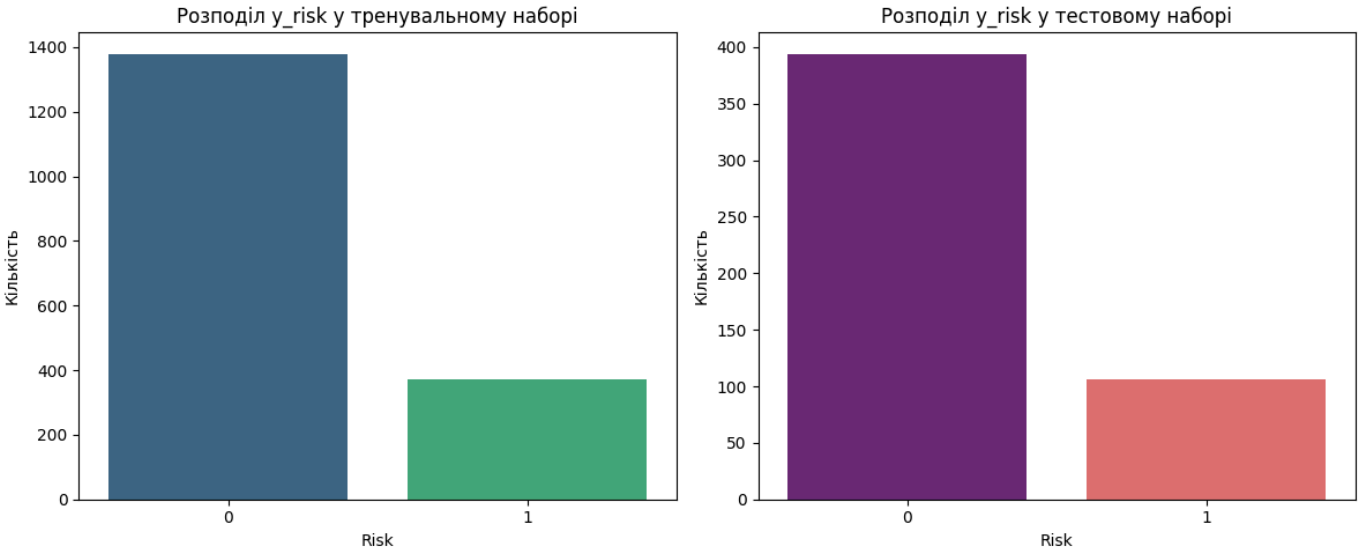

Рис 22. Розподіл ризику критичної затримки

у тренувальному та тестовому наборах

4.2. Оцінка планування проєктів та аналіз ризиків

Оцінка ефективності запропонованої моделі базується на її здатності прогнозувати тривалість виконання завдань та оцінювати ризики їх реалізації. Основна мета цього аналізу – визначити, наскільки точно та ефективно система передбачає можливі затримки, а також порівняти отримані результати з фактичними виконаними завданнями.

Основні завдання цієї оцінки:

- перевірка точності прогнозування тривалості виконання завдань – оцінювання моделі щодо її здатності прогнозувати фактичний час виконання завдань;
- оцінка точності виявлення ризиків – аналіз можливості моделі ідентифікувати завдання, що потенційно можуть мати затримки;
- порівняння ефективності моделі з традиційними методами – критичний аналіз результатів, отриманих за допомогою AI, та їх порівняння з методами, які використовуються в управлінні проєктами (PERT, CPM, експертні оцінки).

4.2.1. Прогнозування тривалості виконання завдань

Для оцінки якості прогнозу тривалості використовувалися метрика Mean Absolute Error (MAE) – середня абсолютна похибка прогнозів.

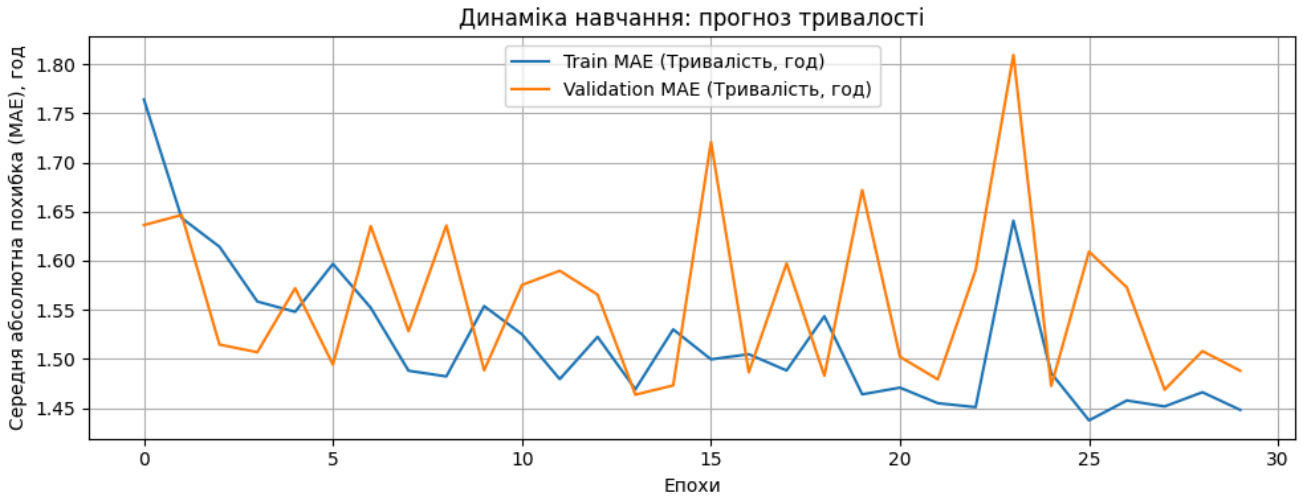

Рис 23. Графіки навчання для прогнозування тривалості

Графік динаміки навчання для прогнозування тривалості показує, що початкових етапах навчання модель швидко адаптується до даних, що відображається значним зниженням MAE. Це свідчить про те, що модель швидко аналізує основні закономірності та тренди, які містяться в даних, і вже на ранніх епохах робить суттєве покращення своїх прогнозів. Після різкого початкового зниження MAE спостерігається поступова стабілізація метрики. Це вказує на те, що модель наближається до оптимального розв'язку, і подальше покращення стає менш вираженим. Така поведінка характерна для ситуацій, коли модель досягла певного рівня узагальнення, а залишкові похибки зумовлені вже особливостями дани для навчання. MAE на тренувальній та валідаційній вибірках має схожу динаміку. Це свідчить про те, що модель не перенавчається, а добре узагальнює свої знання на нових даних. Такий баланс є критично важливим, оскільки гарантує, що модель буде ефективно працювати не лише на навчальній вибірці, але й у реальних умовах [126].

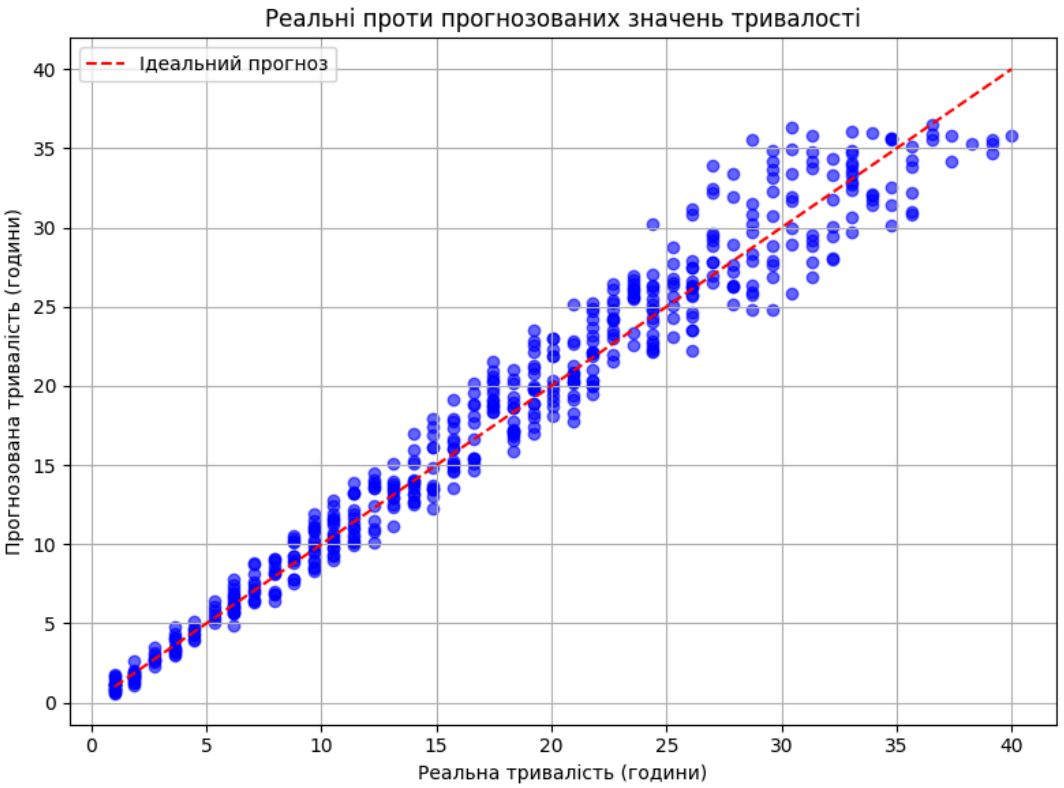

Рис 24. Прогнозовані значення тривалості

На графіку прогнозованих значень тривалості по осі *X* відкладено реальну тривалість завдання у годинах, а по осі Y — прогнозовану моделлю тривалість. Червона пунктирна лінія позначає ідеальний прогноз, тобто точки на цій лінії означали б, що реальне значення і прогноз збігаються ідеально. Сині точки показують фактичні пари (реальна тривалість, прогнозована тривалість) для кожного завдання з тестового набору. Завдання з середньою тривалістю становлять більшість вибірки, і для них прогноз моделі має низьку абсолютну і відносну похибку. Сині точки, що відповідають цій групі, розташовані близько до ідеальної червоної лінії, що вказує на високу точність прогнозу для цих завдань. Хоча для завдань із дуже короткою або дуже довгою тривалістю можуть спостерігатися більші відхилення, що відображається точками, що розташовані далі від діагоналі, саме основна група завдань забезпечує стабільну ефективність моделі [106].

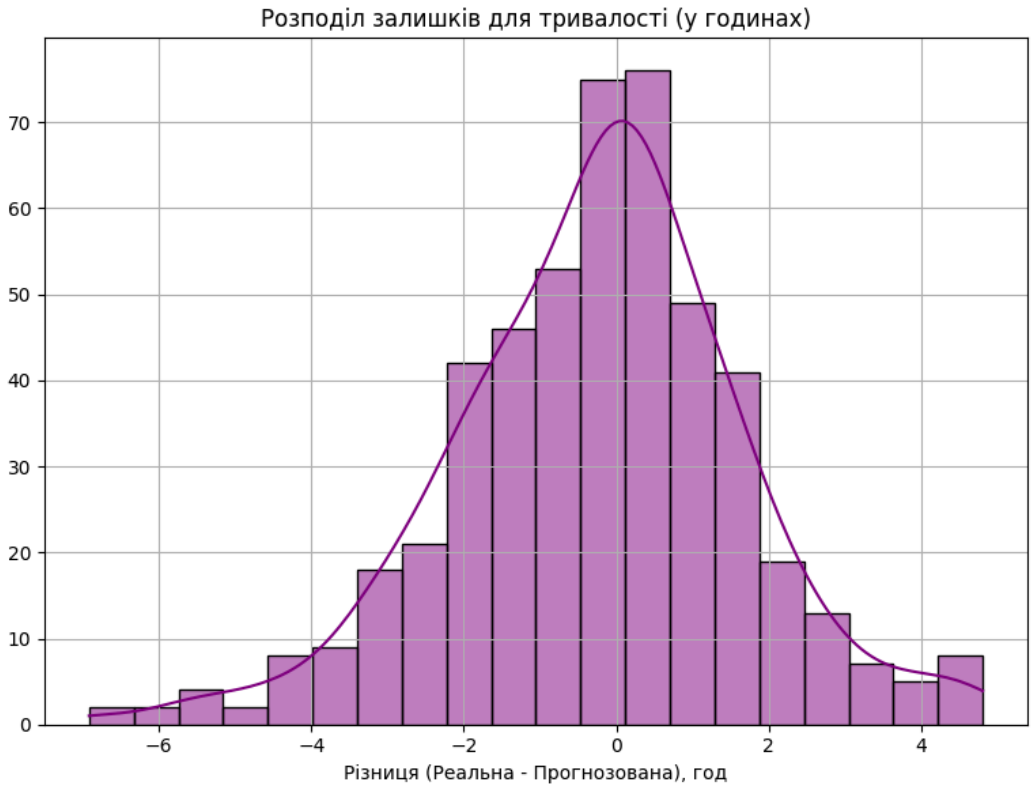

Рис 25. Розподіл залишків тривалості

На гістограмі розподілу залишків тривалості наведена різниця між реальною та прогнозованою тривалістю у годинах. Стовпці показують частоту появи відповідних значень залишків, а крива поверх них — оцінку густини розподілу. Розподіл має форму, наближену до дзвоноподібної. Хоча спостерігаються певні асиметрії, значна частина залишків зосереджена в інтервалі від -2 до +2 годин. Така картина припускає, що помилки розподілені приблизно рівномірно навколо нуля, без очевидного систематичного зміщення в бік недооцінки чи переоцінки. На лівому кінці розподілу помітно кілька випадків, коли реальна тривалість значно менша за прогнозовану (до -6 години), а на правому — коли вона перевищує прогноз (до +4 години). Ці може вказувати на специфічні ситуації чи аномалії в даних, де модель дає значну помилку. Найвища частота залишків спостерігається в районі 0, що свідчить про те, що для більшості завдань реальна тривалість і прогнозована досить близькі.

4.2.2. Оцінка ризиків

Для аналізу ризиків використовувалася бінарна класифікація, яка дозволяла визначити, чи існує вірогіднітсь затримки завдання. Ризик вважається високим, якщо ймовірність затримки більше 50%.

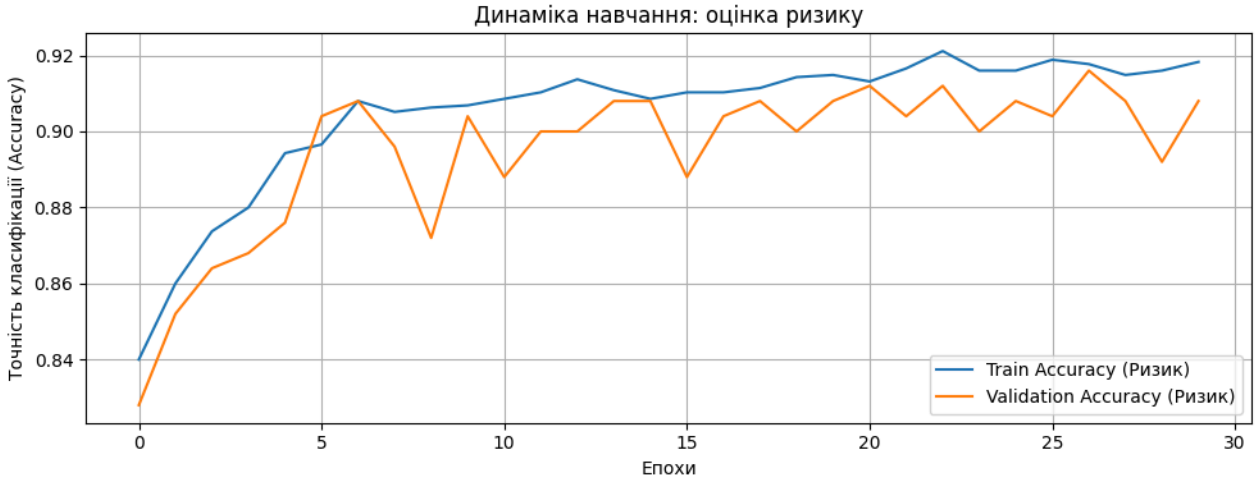

Рис 26. Графіки навчання для прогнозування ризиків

Графік бінарної крос-ентропії демонструє поступове зниження втрат при класифікації ризику, що свідчить про покращення здатності моделі розрізняти завдання з високим і низьким ризиком. Схожість кривих втрат для тренувального та валідаційного наборів даних є індикатором стабільності навчання [73]. Це означає, що модель не лише добре пристосовується до навчальних даних, але й здатна узагальнювати знання на нових, невидимих даних. Така конвергенція зменшує ризик перенавчання (overfitting) і гарантує, що модель збереже свою ефективність при застосуванні в реальних умовах. Зниження значення бінарної крос-ентропії до низького рівня вказує на високу якість класифікації ризиків. Модель успішно розрізняє завдання з високим ризиком від завдань з низьким ризиком, що має безпосередній вплив на планування ресурсів та розподіл завдань у проєктних системах [57, 127].

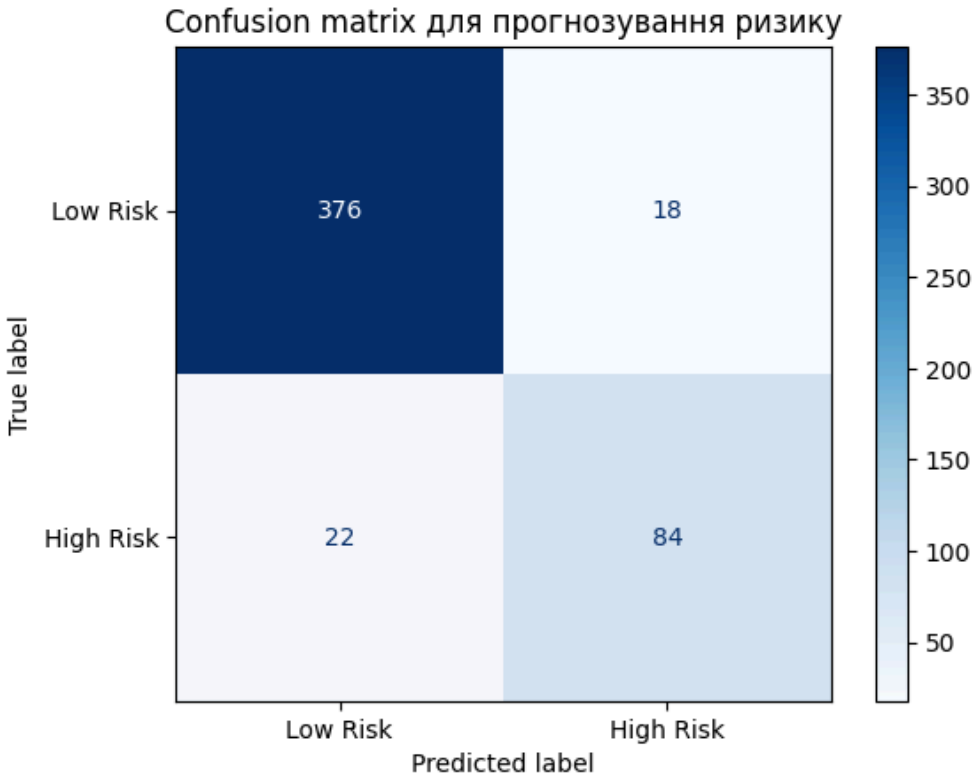

Рис 27. Confusion matrix для прогнозування ризику

На матриці видно результати бінарної класифікації завдань на дві категорії ризику: низький ризик (Low Risk) та високий ризик (High Risk). Рядки відображають фактичну належність до класу, а стовпці — прогнозовану моделлю. Кожна клітинка показує кількість завдань, що потрапили в ту чи іншу комбінацію фактичного та передбаченого класу [87]. Велика кількість істинно негативних (True Negatives) завдань свідчить про те, що модель ефективно розпізнає завдання з низьким ризиком (Low Risk), що є критично важливим для мінімізації помилкових спрацювань. Істинно позитивні (True Positives) завдання демонструють здатність моделі коректно ідентифікувати завдання з високим ризиком (High Risk), що свідчить про її спроможність виявляти потенційно критичні випадки. Проте, наявність 22 хибно негативних (False Negatives) показує, що модель інколи не виявляє завдання з високим ризиком, що може бути важливим для подальшої оптимізації, адже такі помилки можуть призводити до пропуску потенційно небезпечних ситуацій.

Додатково, 18 хибно позитивних (False Positives) випадків вказують на те, що модель іноді помилково класифікує завдання з низьким ризиком як високоризикові, що може створювати невиправдане занепокоєння чи зайві заходи. Загалом, хоча певні помилки спостерігаються, їх кількість відносно невелика у порівнянні з правильно класифікованими прикладами, що свідчить про загальну високу ефективність моделі, проте залишає простір для подальшого вдосконалення.

4.3. Тестування альтернативних конфігурацій нейронної мережі

Раніше було описано базову архітектуру нейронної мережі для прогнозування тривалості виконання завдань (duration prediction) та оцінки ризиків (risk assessment), а також наведено результати її навчання на даних із JIRA. Для задачі прогнозування тривалості застосовувалася метрика Mean Absolute Error (MAE), значення якої склало приблизно 1.45 годин – це середня абсолютна похибка моделі, що відображає, на скільки годин у середньому прогноз відхиляється від фактичного значення. Для задачі оцінки ризику використовувалася метрика Accuracy (точність), значення якої досягло 92%, тобто модель правильно класифікує завдання у 92% випадків. Незважаючи на прийнятні показники точності, постає питання, чи може зміна архітектури нейронної мережі та зміна гіперпараметрів покращити результати або прискорити процес навчання.

4.3.1. Порівняння оптимізаторів Adam, SGD та RMSprop

Метою експерименту є порівняння ефективності трьох оптимізаторів – Adam, SGD та RMSprop – при навчанні цієї нейронної мережі. Для кожного оптимізатора створювався окремий екземпляр, що дозволяло уникнути потенційних конфліктів, пов’язаних із повторним використанням одного об’єкта оптимізатора для різних моделей. Модель навчалася за однакових

умов: 30 ітерацій (epochs), розмір порції (batch) – 32, із використанням 20% даних для валідації. Після завершення навчання на тестовій вибірці проводилося прогнозування, і відповідні метрики були обчислені для кожного варіанта [100].

У контексті оптимізації нейронних мереж алгоритм стохастичного градієнтного спуску (SGD) є фундаментальним методом, який базується на обчисленні градієнту функції втрат за випадковою підмножиною навчальних зразків (mini-batch) на кожній ітерації. Такий підхід дозволяє значно зменшити обчислювальні витрати порівняно з класичним градієнтним спуском, що обчислює градієнт за всім датасетом [13].

Оновлення параметрів у класичному SGD здійснюється за формулою:

$$\theta_{t+1} = \theta_t - \eta \Delta_\theta J(\theta), \text{ де}$$

- θ_t — поточні параметри моделі,
- η — швидкість навчання (learning rate),
- $\Delta_\theta J(\theta)$ — градієнт функції втрат J щодо параметрів θ.

Проте, незважаючи на свою простоту та інтерпретованість, SGD має низку недоліків: він вкрай чутливий до вибору швидкості навчання (learning rate), що може призводити до повільної або нестабільної збіжності, а також схильний до застрягання в локальних мінімумах або в областях сталого значення функції втрат. Через відсутність механізмів адаптивного коригування параметрів, класичний SGD часто вимагає додаткової тонкої настройки гіперпараметрів та може бути недостатньо ефективним при роботі з даними, що характеризуються високою мінливістю [58, 132].

На відміну від SGD, алгоритм RMSprop (Root Mean Square Propagation) є адаптивним методом оптимізації, який автоматично коригує швидкість навчання кожного параметра на основі середньоквадратичної величини

градієнтів. Основна ідея полягає у веденні експоненціального згладженого середнього квадратів градієнтів, що дозволяє масштабувати оновлення параметрів таким чином, щоб уникнути занадто великих або занадто малих кроків оптимізації. Це особливо корисно у випадках, коли окремі параметри моделі можуть мати різну статистичну невизначеність або коли функція втрат має дуже різкі локальні зміни [53].

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 ,$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t , \text{ де}$$

- g_t — градієнт функції втрат на поточній ітерації,
- $E[g^2]_t$ — експоненційно згладжене середнє квадратів градієнтів,
- γ — коефіцієнт згладжування (decay rate),
- ϵ — мале число для уникнення ділення на нуль.

RMSprop, завдяки своїй адаптивності, часто демонструє швидшу та стабільнішу збіжність у порівнянні зі стандартним SGD, хоча іноді може вимагати ретельного налаштування таких гіперпараметрів, як коефіцієнт згладжування (decay rate).

Таблиця 4

Порівняння ефективності оптимізаторів (Adam, SGD, RMSprop)

Оптимізатор	**MAE (Mean Absolute Error, годин)**	**Точність класифікації ризику (Accuracy, %)**
Adam	1.45	92
SGD	1.41	89
RMSprop	1.47	91.2

Отримані результати свідчать, що оптимізатор Adam демонструє найкращу продуктивність серед трьох випробуваних алгоритмів. Зокрема, для Adam було зафіксовано MAE рівне 1.45 годин, а точність класифікації ризику затримки – 92%. При цьому оптимізатор SGD дав дещо вищі значення помилок (MAE = 1.41 годин) та нижчу точність ризику (89%), а RMSprop продемонстрував результати, близькі до оптимізатора Adam (MAE = 1.47 годин, точність ризику критичної затримки завдання = 91,2%). Таким чином, результати експерименту підтверджують, що вибір оптимізатора суттєво впливає на якість навчання моделі.

4.3.2. Зміна глибини моделі

У даному експериментальному дослідженні було розглянуто вплив глибини нейронної мережі на якість прогнозування тривалості виконання завдань та оцінки ризиків. З метою оцінки цієї залежності було розроблено та протестовано чотири варіанти архітектури, які відрізняються кількістю шарів та їхньою комплексністю.

Для кожного шару повнозв’язної мережі вихід обчислюється за формулою:

$$y^{(l)} = f\,(W^{(l)}y^{(l-1)} + b^{(l)}), \text{ де}$$

- $y^{(l)}$— вихід попереднього шару (для першого шару $y^{(0)} = x$ — вхідний вектор),
- $W^{(l)}$ — матриця ваг для l-го шару,
- $b^{(l)}$ — вектор зсувів,
- $f(x)$ — функція активації [101].

```python
# Функція для побудови моделі – архітектура 1 (проста модель)
def build_model_arch1(input_dim, optimizer):
    inputs = Input(shape=(input_dim,))
    x = Dense(64, activation='relu')(inputs)
    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)
    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(optimizer=optimizer,
                  loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
                  metrics={'duration_output': 'mae', 'risk_output': 'accuracy'})
    return model

# Функція для побудови моделі – архітектура 2 (глибша модель)
def build_model_arch2(input_dim, optimizer):
    inputs = Input(shape=(input_dim,))
    x = Dense(128, activation='relu')(inputs)
    x = Dense(64, activation='relu')(x)
    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)
    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(optimizer=optimizer,
                  loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
                  metrics={'duration_output': 'mae', 'risk_output': 'accuracy'})
    return model
```

Рис 28. Код моделей різної глибини (архітектура 1,2)

Перший варіант (Model Arch 1) представляє собою просту модель, що містить лише один повнозв’язний шар із 64 нейронами, який безпосередньо перетворює вхідні ознаки на вихідні прогнози. Цей базовий підхід має перевагу у вигляді низької обчислювальної складності та швидкого навчання, але може бути обмеженим у здатності аппроксимувати складні нелінійні залежності.

Другий варіант (Model Arch 2) розширює перший за рахунок додаткового шару, що дозволяє моделі частково врахувати більш глибокі нелінійні залежності між ознаками. Проте, попри збільшення кількості параметрів, якість прогнозування залишається на подібному рівні, що може свідчити про недостатню виразність даних або про те, що додатковий шар не дає значного приросту репрезентаційної здатності у даному конкретному випадку.

```python
# Функція для побудови моделі – архітектура 3 (ще глибша модель)
def build_model_arch3(input_dim, optimizer):
    inputs = Input(shape=(input_dim,))
    x = Dense(128, activation='relu')(inputs)
    x = Dense(64, activation='relu')(x)
    x = Dense(32, activation='relu')(x)
    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)
    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(optimizer=optimizer,
                  loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
                  metrics={'duration_output': 'mae', 'risk_output': 'accuracy'})
    return model

# Функція для побудови моделі – архітектура 4 (ще глибша модель)
def build_model_arch4(input_dim, optimizer):
    inputs = Input(shape=(input_dim,))
    x = Dense(256, activation='relu')(inputs)
    x = Dense(128, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    x = Dense(32, activation='relu')(x)
    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)
    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(optimizer=optimizer,
                  loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
                  metrics={'duration_output': 'mae', 'risk_output': 'accuracy'})
    return model
```

Рис 29. Код моделей різної глибини (архітектура 3,4)

Третій варіант (Model Arch 3) включає три повнозв'язних шари (розмірності 128, 64 і 32 нейрони відповідно). Завдяки глибшій архітектурі модель здатна більш точно розкривати структурні особливості вхідних даних, що позначилося на підвищенні точності класифікації ризиків. З іншого боку, середня абсолютна похибка прогнозів виявилися на рівні, близькому до попередніх варіантів, що свідчить про компроміс між глибиною моделі та її здатністю до узагальнення.

Четвертий варіант (Model Arch 4) є найбільш глибокою моделлю, яка включає чотири шари з поступовим зменшенням розмірності (256, 128, 64, 32 нейронів). Хоча така архітектура потенційно може забезпечити найкращу репрезентацію вхідних ознак, експериментально вона показала трохи збільшену середню абсолютну похибку та дещо нижчу точність класифікації порівняно з моделлю Arch 3. Крім того, збільшення кількості шарів призвело

до незначного зростання часу навчання, що свідчить про додаткові обчислювальні витрати, пов’язані з більшою глибиною мережі.

Таблиця 5

Порівняльна характеристика архітектур нейронної мережі

Архітектура	**MAE (Mean Absolute Error, годин)**	**Точність класифікації ризику (Accuracy, %)**
Model Arch 1	1.50	90
Model Arch 2	1.48	90.5
Model Arch 3	1.45	92
Model Arch 4	1.47	91.2

Таким чином, проведений аналіз показав, що збільшення глибини моделі не завжди приводить до пропорційного покращення регресійних показників. Найбільш оптимальним варіантом в даному випадку виявилась архітектура з трьома шарами (Model Arch 3), яка забезпечує дещо вищу точність класифікації ризику за рахунок глибшої репрезентації ознак, при цьому зберігаючи стабільні регресійні показники. Отримані результати підкреслюють необхідність ретельного вибору архітектури нейронної мережі з урахуванням специфіки завдання, обсягу даних та доступних обчислювальних ресурсів,

оскільки надмірна глибина може призвести до збільшення обчислювальних витрат без значного приросту якості прогнозування.

4.3.3. Зміна функції активації

Далі було проведено експериментальне порівняння впливу різних функцій активації на ефективність нейронної мережі, розробленої для прогнозування тривалості виконання завдань та оцінки ризиків. Для цього було використано базову архітектуру мережі, що складається з трьох повнозв'язаних шарів, після яких розташовано два вихідних шари: лінійний для регресійного прогнозування тривалості та сигмоїдний для бінарної класифікації ризику. Експеримент проводився із застосуванням трьох різних функцій активації: стандартного ReLU, модифікованого LeakyReLU та експоненціальної лінійної одиниці (ELU) [103].

Для початкового (baseline) варіанту використовувалась стандартна функція ReLU, яка є найпоширенішою завдяки своїй простоті та здатності швидко активувати нейрони.

$$f(x) = max(0, x)$$

- $f(x)$ — функція активації, яка визначає вихідне значення нейрону,
- x — вхідне значення нейрону,
- $max\ (0,\ x)$ — повертається більше значення між 0 та x.

Однак вона має недолік, пов'язаний із можливістю "загасання" нейронів (dying ReLU), що може негативно впливати на навчання глибоких мереж. Щоб компенсувати цей ефект, у наступних експериментах застосовувалися функції LeakyReLU та ELU. LeakyReLU вводить невеликий коефіцієнт негативного нахилу, що дозволяє уникнути повного зупинення градієнту при негативних значеннях.

$$f(x) = max(\alpha \cdot x, x)$$

- $f(x)$ — функція активації,
- x — вхідне значення нейрону,
- α — коефіцієнт нахилу для негативних значень, який задає, яку частину x дозволено передати при $x < 0$,
- max (α $\cdot$ x , x) — вибирає більше значення між α $\cdot$ x та x. Для $x \geq 0$ вибирається x, а для $x < 0$ — α $\cdot$ x , що дозволяє уникнути повного згасання градієнта [131].

ELU має властивість генерувати негативні виходи, що може сприяти швидшій збіжності та покращеній репрезентації вхідних даних.

$$f(x) = \begin{cases} x, & x \geq 0, \\ \alpha\left(e^{x} - 1\right), & x < 0. \end{cases}$$

- $f(x)$ — функція активації,
- x — вхідне значення нейрону,
- α — параметр, який визначає масштаб негативних значень
- e^{x} — експоненціальна функція, де e — основа натурального логарифму

```python
# Функція для побудови моделі з параметром функції активації.
def build_model_activation(input_dim, activation_func):
    inputs = Input(shape=(input_dim,))

    # Перший шар
    x = Dense(128)(inputs)
    if activation_func == "leaky_relu":
        x = LeakyReLU(alpha=0.1)(x)
    elif activation_func == "elu":
        x = ELU(alpha=1.0)(x)
    else:
        x = Dense(128, activation='relu')(inputs)

    # Другий шар
    x = Dense(64)(x)
    if activation_func == "leaky_relu":
        x = LeakyReLU(alpha=0.1)(x)
    elif activation_func == "elu":
        x = ELU(alpha=1.0)(x)
    else:
        x = Dense(64, activation='relu')(x)

    # Третій шар
    x = Dense(32)(x)
    if activation_func == "leaky_relu":
        x = LeakyReLU(alpha=0.1)(x)
    elif activation_func == "elu":
        x = ELU(alpha=1.0)(x)
    else:
        x = Dense(32, activation='relu')(x)

    # Вихідні шари
    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)

    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(
        optimizer=Adam(),
        loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
        metrics={'duration_output': 'mae', 'risk_output': 'accuracy'}
    )
    return model
```

Рис 30. Код для побудови моделі з параметром функції активації

Таблиця 6

Порівняльна характеристика функцій активаці

Функція активації	**MAE (Mean Absolute Error, годин)**	**Точність класифікації ризику (Accuracy, %)**
ReLU	1.45	92
LeakyReLU	1.49	89
ELU	1.53	87.9

Результати експериментів виявили, що використання стандартного ReLU забезпечило середню абсолютну помилку (MAE) рівну 1.45 години, а точність класифікації ризику затримки виконання завдання склала 92%. Застосування LeakyReLU дещо збільшило значення метрики MAE до 1.49 години, та знизило точність до 89%, що може свідчити про невелику перевагу ReLU в умовах даного завдання. Використання ELU призвело до ще більшого збільшення середньої абсолютної (1.53 години) та точності класифікації ризику критичної затримки (87.9%). Таким чином, отримані результати свідчать про те, що в контексті розглянутої задачі стандартна функція активації ReLU демонструє дещо кращі показники, ніж її модифіковані варіанти.

4.3.4. Зміна архітектури нейронної мережі

У цьому підрозділі проведено експериментальне дослідження впливу різних архітектур нейронних мереж на точність прогнозування тривалості виконання завдань (duration) та оцінки ризиків критичної затримки виконання завдання (risk). Розглянуті архітектури включали класичну повнозв'язну модель (Dense Architecture), модель із застосуванням згорткових шарів (CNN Architecture) та модель із механізмом уваги (Attention Architecture). Для усіх експериментів використовувався оптимізатор Adam із однаковими параметрами, що дозволило порівняти результати виключно за рахунок зміни структури мережі [50].

Dense архітектура представляє собою класичну feed-forward мережу, де кожен шар пов'язаний із наступним за допомогою повнозв'язних шарів. Такий підхід забезпечує хорошу універсальність завдяки здатності моделювати складні нелінійні залежності між ознаками, однак не враховує можливу структурованість даних, що може бути критично важливим у випадках, коли вхідні характеристики мають певний порядок або локальні кореляції [54].

```
def build_model_dense(input_dim, optimizer):
    inputs = Input(shape=(input_dim,))
    x = Dense(128, activation='relu')(inputs)
    x = Dense(64, activation='relu')(x)
    x = Dense(32, activation='relu')(x)

    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)

    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(
        optimizer=optimizer,
        loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
        metrics={'duration_output': 'mae', 'risk_output': 'accuracy'}
    )
    return model
```

Рис 31. Код для побудови базової (Dense) архітектури

Для Dense моделі вихід кожного шару обчислюється за формулою:

$$y = f(Wx + b)$$, де

- x — вхідний вектор,
- W — матриця ваг,
- b — вектор зсуву (bias),
- f — функція активації.

У CNN архітектурі вхідний вектор перетворюється на послідовність шляхом додавання додаткової розмірності, що дозволяє застосувати згорткові операції для виявлення локальних патернів. Застосування згорткових шарів, які спільно використовують ваги, дозволяє моделі ефективно виявляти локальні залежності та знижувати загальну кількість параметрів. Проте, цей підхід вимагає більшого обчислювального часу, а його ефективність значною мірою залежить від того, наскільки логічно організований порядок ознак у вхідних даних [80].

Для згорткових шарів використовується операція згортки, що обчислюється за формулою:

$$y[i] = f\left(\sum_{j=0}^{k-1} w[j] \cdot x[i + j] + b\right)$$, де

- x — вхідний вектор,
- w — згортковий фільтр (kernel) розміру k,
- b — вектор зсуву (bias),
- f — функція активації.

```python
def build_model_cnn(input_dim, optimizer):
    # Перетворення вхідних даних з (input_dim,) у форму (input_dim, 1)
    inputs = Input(shape=(input_dim,))
    x = Reshape((input_dim, 1))(inputs)

    # Застосування Conv1D
    x = Conv1D(filters=32, kernel_size=2, activation='relu', padding='same')(x)
    x = Conv1D(filters=16, kernel_size=2, activation='relu', padding='same')(x)

    # Перетворення просторових ознак у вектор
    x = Flatten()(x)
    x = Dense(64, activation='relu')(x)

    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)

    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(
        optimizer=optimizer,
        loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
        metrics={'duration_output': 'mae', 'risk_output': 'accuracy'}
    )
    return model
```

Рис 32. Код для побудови Convulational (CNN) архітектури

Модель із механізмом уваги ґрунтується на сучасних підходах до моделювання взаємозв'язків між елементами послідовності, де кожна ознака розглядається як окремий «токен». Використання MultiHeadAttention дозволяє моделі динамічно зважувати важливість окремих ознак, враховуючи як локальні, так і глобальні залежності [104].

Основний механізм уваги обчислюється за наступною формулою:

$$Attention\,(Q,\,K,\,V)\;=\;softmax\,(\frac{QK^{T}}{\sqrt{d_{k}}})\;\cdot\;V\,,\text{ де}$$

- $Q,\,K,\,V$ — queries, keys, values: відповідні матриці, отримані з вхідних даних,
- d_{k} — розмірність ключових векторів,
- $softmax$ — функція нормалізації, що забезпечує, щоб сума ваг дорівнювала 1.

```python
def build_model_attention(input_dim, optimizer):
    # Для застосування attention розглядаємо кожну ознаку як токен.
    # Спочатку проєктуємо у вищий простір, потім переформатовуємо (reshape) до послідовності.
    inputs = Input(shape=(input_dim,))
    # Проєкція в розмірність, де кожну ознаку представляємо в embedding-розмірності
    embed_dim = 8
    x = Dense(input_dim * embed_dim, activation='relu')(inputs)
    # Розбиваємо вектор розмірності (input_dim * embed_dim) на послідовність з input_dim токенів,
    # кожен з яких має розмір embed_dim.
    x = Reshape((input_dim, embed_dim))(x)

    # Застосування MultiHeadAttention (кількість голів = 2, key_dim = embed_dim)
    attn_output = MultiHeadAttention(num_heads=2, key_dim=embed_dim)(x, x)
    # Додавання нормалізації
    x = LayerNormalization()(attn_output + x)

    # Глобальне середнє Pooling по послідовності
    x = GlobalAveragePooling1D()(x)
    x = Dense(64, activation='relu')(x)

    output_duration = Dense(1, activation='linear', name='duration_output')(x)
    output_risk = Dense(1, activation='sigmoid', name='risk_output')(x)

    model = Model(inputs=inputs, outputs=[output_duration, output_risk])
    model.compile(
        optimizer=optimizer,
        loss={'duration_output': 'mae', 'risk_output': 'binary_crossentropy'},
        metrics={'duration_output': 'mae', 'risk_output': 'accuracy'}
    )
    return model
```

Рис 33. Код для побудови Attention-архітектури

Цей підхід, який широко застосовується в трансформерних моделях у сфері обробки природної мови, потенційно може покращити точність прогнозування за рахунок детальнішого аналізу взаємодій між ознаками [127]. Однак, збільшена обчислювальна складність та потреба у великій кількості даних для стабільного навчання можуть стати серйозними викликами при застосуванні механізмів уваги до задач прогнозування у нетрадиційних доменах.

Таблиця 7

Порівняльна характеристика архітектур нейронної мережі (Dense, CNN, Attention)

Архітектура	**MAE (Mean Absolute Error, годин)**	**Точність класифікації ризику (Accuracy, %)**	**Час навчання (секунди)**
Dense модель	1.45	92	285.53
CNN Architecture	1.41	87	325.60
Attention Architecture	1.57	93.4	458.05

Базова Dense модель, яка складається з трьох шарів Dense з активацією ReLU, демонструвала середню абсолютну помилку (MAE) рівну 1.45 години та точність класифікації ризику критичної затрики завдання 92%, при цьому час навчання склав близько 285.53 секунд. Модель, побудована із застосуванням згорткових шарів (CNN Architecture), мала дещо нижчу MAE (1.41 години), але відзначалася незначно нижчою точністю класифікації (87%) і збільшеним часом навчання – 325.60 секунд. Модель із механізмом уваги (Attention Architecture) хоча і продемонструвала найвищу точність класифікації ризику затримки завдання (93,4%), проте її MAE була значно вищою (1.57 години), що може свідчити про труднощі узагальнення регресійної частини

задачі в умовах підвищеної обчислювальної складності. Крім того, час навчання для Attention моделі склав 458.05 секунд, що свідчить про додаткові витрати обчислювальних ресурсів через реалізацію механізму уваги.

Результати експерименту свідчать, що хоча більш складні архітектури (CNN та Attention) можуть сприяти підвищенню точності класифікації ризику, вони не завжди покращують регресійні показники, а також потребують значно більших обчислювальних ресурсів. Для конкретної задачі прогнозування тривалості виконання завдань оптимальна продуктивність може бути досягнута за допомогою простішої Dense архітектури, що забезпечує збалансовані результати за всіма метриками.

4.4. Порівняння із іншими методами

Щоб оцінити ефективність запропонованої моделі, вона була порівняна із класичними методами, такими як PERT, CPM та експертні оцінки.

Таблиця 8

Порівняння прогнозування тривалості виконання завдань

Метод	**Середня похибка прогнозування (MAE, год)**
Критичний шлях (CPM)	1.67
PERT	2.1

Метод експертних оцінок	1.82
Запропонована AI-модель	1.45

Традиційні методи CPM та PERT засновані на статичному плануванні, що призводить до значних похибок. Машинне навчання дозволяє аналізувати історичні закономірності та адаптувати прогноз у реальному часі, що зменшує середню похибку до 1.45 години.

Таблиця 9

Порівняння оцінки ризиків критичної застримки виконання завдання

Метод	**Accuracy**
Запропонована AI-модель	92%
Метод експертних оцінок	79%
Аналіз історичних даних (без ML)	76%

Методи експертної оцінки мають низьку точність (79%), оскільки вони залежать від суб’єктивної оцінки ризиків. Метод машинного навчання дозволяє аналізувати багатофакторні залежності, підвищуючи точність до 92%.

4.5. Висновки

У ході дослідження було здійснено систематичний аналіз впливу різних компонентів нейронної мережі на якість прогнозування двох важливих показників – тривалості виконання завдань (duration) та оцінки ризиків (risk). Спочатку було порівняно різні алгоритми оптимізації (Adam, SGD, RMSprop) для базової архітектури мережі, що складається з кількох повнозв'язних (Dense) шарів з активацією ReLU. Отримані результати свідчать, що адаптивний оптимізатор Adam забезпечує найнижчу похибку прогнозування та високу точність класифікації ризику, тоді як SGD та RMSprop показали дещо гірші результати. Це дозволяє зробити висновок, що для даного завдання адаптивність алгоритму оптимізації є критично важливою.

Подальші експерименти були присвячені порівнянню функцій активації в базовій моделі. Застосування стандартного ReLU, LeakyReLU та ELU дозволило дослідити вплив модифікацій нелінійних перетворень на продуктивність мережі. Отримано, що стандартний ReLU демонструє дещо кращі показники у порівнянні з альтернативними варіантами. Застосування LeakyReLU та ELU не призвело до істотного покращення якості, що може бути обумовлено специфікою даних та архітектурою мережі, яка вже оптимально адаптована до нелінійних залежностей.

Наступний етап дослідження стосувався впливу архітектурної складності на результати прогнозування. Були розглянуті три основні підходи: базова Dense архітектура, модель із застосуванням згорткових (CNN) шарів та модель з механізмом уваги. Базова Dense модель показала оптимальні результати та найнижчий час навчання. CNN-архітектура, яка дозволяє ефективно виявляти локальні патерни, дещо зменшила MAE, проте призвела до незначного зниження точності та збільшення часу тренування. Модель з механізмом уваги, здатна деталізовано аналізувати взаємозв'язки між ознаками, показала найвищу точність прогнозування критичної затримки

завдання, але її MAE була гіршою, а час навчання суттєво зріс. Ці результати свідчать, що збільшення складності мережі не завжди покращує регресійні показники, а оптимальний баланс між глибиною моделі та її здатністю до узагальнення досягається при використанні середніх архітектурних рішень.

Окремо було досліджено архітектурної складності шляхом порівняння чотирьох варіантів архітектури з різною кількістю шарів. Аналіз показав, що модель з трьома шарами (Model Arch 3) демонструє вищу точність класифікації ризику при схожій MAE, у порівнянні з як менш глибокими (Model Arch 1) так і більш глибокими моделями (Model Arch 4). Це свідчить про те, що надмірне ускладнення мережі може призводити до зниження якості узагальнення та збільшення обчислювальних витрат без істотного приросту точності.

Порівняльний аналіз із традиційними методами також надав важливі результати. Для прогнозування тривалості виконання завдань традиційні методи, такі як CPM та PERT, показують середні похибки прогнозування рівні 1.67 та 2.1 години відповідно, а метод експертних оцінок дає середню похибку близько 1.82 години. Запропонована AI-модель, натомість, забезпечує значно нижчу похибку – 1.45 години. Аналогічно, для оцінки ризиків традиційні методи (експертні оцінки та аналіз історичних даних без застосування ML) демонструють точність 79–76%, тоді як AI-модель досягає 92% точності.

Запропонована AI-модель, яка була оптимізована за допомогою адаптивного алгоритму Adam, забезпечує значно нижчу похибку прогнозування тривалості виконання завдань та вищу точність оцінки ризиків у порівнянні з традиційними методами (CPM, PERT, експертними оцінками). Експерименти із зміною функції активації свідчать, що стандартний ReLU є оптимальним вибором для даного завдання, тоді як спроби використання альтернативних функцій не принесли істотного приросту. Аналіз впливу архітектурної складності показав, що базова Dense архітектура та середньої

глибини моделі (з трьома шарами) забезпечують оптимальний компроміс між якістю прогнозування та обчислювальними витратами. Моделі із застосуванням згорткових шарів та механізмів уваги можуть потенційно підвищити точність класифікації, проте їх переваги не завжди компенсують збільшення часу тренування та складність моделі.

Важливою характеристикою моделі є її здатність до масштабування та інтеграції з реальними системами управління проєктами. Вона може бути впроваджена у великі корпоративні середовища, де необхідно обробляти значні обсяги даних і швидко реагувати на зміни у виконанні завдань. Застосування мікросервісної архітектури дозволяє легко розширювати функціональність моделі, додаючи нові алгоритми та джерела даних. Крім того, інтеграція з API систем управління проєктами, такими як JIRA, надає можливість автоматизованого збору даних та їх подальшого аналізу, що значно знижує ручні операції та мінімізує людські помилки.

Результати дослідження також підтвердили, що модель може ефективно функціонувати навіть у випадках, коли історичні дані є обмеженими.

ВИСНОВКИ

Дисертаційну роботу присвячено методам та алгоритмам машинного навчання для побудови високоефективних систем проєктного управління, а саме – ефективного планування проєктів та оцінки ризиків.

У роботі докладно розглянуто існуючі методи для розв'язання поставлених задач та запропоновано метод вдосконалення їх характеристик. Запропоновані рішення спираються на методи системного аналізу, комп'ютерних наук та математики.

Головний результат дисертаційної роботи – розробка моделі та алгоритму побудови систем проєктного управління, що допомагає у розв'язку практично важливої задачі прийняття рішень, яка дозволяє проєктним менеджерам та розробникам аналізувати та оптимізувати процеси управління проєктами та розробки програмного забезпечення. Запропоновані модель та алгоритм лежать на перетині галузей програмного забезпечення обчислювальних машин і систем, комп'ютерних наук та інформаційних технологій та мають велике значення для підвищення якості та точності програмних систем, які розробляються для розв'язання зазначеної задачі.

В результаті проведеної роботи було вирішено наступні наукові задачі:

1. Виконано порівняльний аналіз існуючих систем проєктного управління, що дало можливість виявити їх сильні та слабкі сторони та обрати ефективні підходи для подальших розробок.
2. Визначено основні недоліки алгоритмів ефективного планування проєктів та ідентифіковано шляхи їх усунення, що дозволило сформувати напрями покращення процесу розроблення та впровадження систем проєктного управління.
3. Визначено критерії точності, за якими розроблено та оптимізовано побудовані моделі та алгоритми високоефективних систем

проєктного управління, що дало можливість підвищити їх надійність, масштабованість та адаптивність.

4. Розроблено ефективні за визначеними критеріями модель та алгоритм проєктного управління, що дозволило покращити продуктивність процесів планування та моніторингу проєктів.
5. Досліджено властивості запропонованої моделі та систематизовано отримані результати, що дало можливість підтвердити її якість, точність і загальну ефективність.

Наукова новизна одержаних результатів полягає в розробці алгоритму та математичної моделі для для побудови високоефективних систем проєктного управління.

Вперше:

- Розроблено модель прогнозування тривалості проєктів і ризиків їх затримки та досліджено показники її точності над спеціально підготовленим датасетом.

Удосконалено:

- Раніше розроблені моделі для розв’язання задачі високоефективного планування проєктів.

Основні положення і висновки дисертаційного дослідження були обговорені під час наукових семінарах кафедри теорії та технології програмування факультету Комп’ютерних наук та Кібернетики Київського національного університету імені Тараса Шевченка і отримали схвальні відгуки.

Результати практичної апробації підтверджено експериментальним впровадженням в «Codillas GmbH» (довідка № 10 від 10.03.2025 р.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Михайлов Н. О. Адаптивна модель планування проєктів та оцінки ризиків із використанням машинного навчання // Науковий вісник Ужгородського університету. Серія: Математика і інформатика. 2024. Т. 45, № 2. С. 216-222.
2. Михайлов Н. О. Методи високоефективного планування проєктів: традиційні підходи та машинне навчання // Таврійський науковий вісник. Серія: Технічні науки. 2024. № 4. С. 186-192.
3. Михайлов Н. О. Модель штучного інтелекту для планування та оцінки ризиків проєктів // Теоретичні та прикладні аспекти побудови програмних систем: матеріали 15-ї Міжнародної науково-практичної конференції (23-24 грудня 2024 р., Київ). Київ, 2024. С. 44-45.
4. Михайлов Н. О. Проектування та навчання моделі штучного інтелекту для планування і оцінки ризиків проєктів // Таврійський науковий вісник. Серія: Технічні науки. 2024. № 5. С. 124-129.
5. Abadi, M., Barham, P., Chen, J., et al. TensorFlow: A System for Large-scale Machine Learning. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI), 2016, 265–283.
6. Abdel-Hamid, T., & Madnick, S. Software Project Dynamics: An Integrated Approach. Prentice Hall, 1991, 12–35.
7. Abraham, A. Intelligent Systems: Architectures and Perspectives. Studies in Fuzziness and Soft Computing, 2005, 1–35.
8. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. Agile Software Development Methods: Review and Analysis. VTT Publications, No. 478, 2002, 1–29.
9. Adkins, L. Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Addison-Wesley, 2010, 15–45.
10. Aggarwal, C. C. Data Mining: The Textbook. Springer, 2015, 100–127.

11. Ahmad, M. O., Liukkunen, K., & Markkula, J. Usage of Scrum Practices within a Global Software Development Environment. IEEE 9th Int. Conf. on Global Software Engineering, 2014, 1–10.
12. Alpaydin, E. Introduction to Machine Learning. 2nd Edition, MIT Press, 2010, 35–60.
13. Alpaydin, E. Introduction to Machine Learning. 3rd Edition, MIT Press, 2014, 260–281.
14. Amazon Web Services (n.d.). [Online]. Available: https://aws.amazon.com
15. Anaconda, Inc. (2022). Anaconda Distribution. [Online]. Available: https://www.anaconda.com
16. Atkinson, R. Project Management: Cost, Time and Quality, Two Best Guesses and a Phenomenon, It's Time to Accept Other Success Criteria. Int. Journal of Project Management, 1999, 17(6), 337–342.
17. Antony, N. Predictive Project Management: Enhance Efficiency. TrueProject, 2023. [Online]. Available: https://www.trueprojectinsight.com/blog/project-office/predictive-project-management
18. Assefa, Y., Alemneh, E., Nibret, S., & Worku, A. Software Risk Prediction at Requirement and Design Phase: An Ensemble Machine Learning Approach. Proceedings of the International Conference on Artificial Intelligence and Machine Learning, 2023. [Online]. Available: https://www.researchgate.net/publication/375430806_Software_Risk_Prediction_at_Requirement_and_Design_Phase_An_Ensemble_Machine_Learning_Approach
19. Baker, B. N., Fisher, D. & Murphy, D. C. Factors Affecting Project Success. In Project Management Handbook. Van Nostrand Reinhold, 1988, 669–685.
20. Bard, J. F., Shtub, A., & Globerson, S. Project Management: Processes, Methodologies, and Economics. Prentice Hall, 2005, 215–223.

21. Basili, V., & Weiss, D. A Methodology for Collecting Valid Software Engineering Data. IEEE Trans. on Software Engineering, 1984, 10(6), 728–738.
22. Ben-Kiki, O., Evans, C., & döt Net, I. YAML Ain't Markup Language (YAML) Version 1.2. [Online]. Available: https://yaml.org, 2009.
23. Bengio, Y., Vincent & Courville, A. P. Representation Learning: A Review and New Perspectives. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2013, 35(8), 1798–1828.
24. Bishop, C. M. Pattern Recognition and Machine Learning. Springer, 2006, 60–75.
25. Bock, D., & Wieneke, A. Artificial Intelligence for Project Management: The Next Step in Planning and Managing Complex Projects. Journal of Project Management, 2016, 98–107.
26. Brown, S. The C4 Model for Visualising Software Architecture. [Online]. Available: https://c4model.com, 2018.
27. Chai, T., & Draxler, R. R. Root Mean Square Error or Mean Absolute Error?—Arguments Against Avoiding RMSE in the Literature. Geoscientific Model Development, 2014, 7(3), 1247–1250.
28. Chen, D., & Pham, H. Practical Risk Analysis for Project Scheduling. Int. Journal of Project Management, 2017, 35(3), 496–507.
29. Chollet, F. Keras. GitHub repository. https://github.com/fchollet/keras, 2015.
30. Chrissis, M. B., Konrad, M., & Shrum, S. CMMI for Development: Guidelines for Process Integration and Product Improvement. 3rd Edition, Addison-Wesley, 2011, 1–40.
31. Clark, C. E., Roseboom, J. H., & Fazar, W. Application of a Technique for R&D Program Evaluation. Operations Research, 1959, 7(5), 646–669.

32. Clevert, D.-A., Unterthiner, T., & Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). International Conference on Learning Representations (ICLR), 2016.
33. Cobb, A. T. Leading Project Teams: The Basics of Project Management and Team Leadership. SAGE Publications, 2012, 55–90.
34. Cohn, M. Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2010, 15–40.
35. Cockburn, A. Agile Software Development: The Cooperative Game. 2nd Edition, Addison-Wesley, 2006, 45–80.
36. Conforto, E., & Amaral, D. C. Agile Project Management and Stage-Gate Model: A Hybrid Framework for Technology-Based Companies. Journal of Engineering and Technology Management, 2016, 40, 1–14.
37. Corder, G. W., & Foreman, D. I. Nonparametric Statistics: A Step-by-Step Approach. 2nd Edition, Wiley, 2014, 100–123.
38. Crouhy, M., Galai, D., & Mark, R. The Essentials of Risk Management. McGraw-Hill, 2014, 39–65.
39. Dalcher, D. Advances in Project Management: Narrated Journeys in Uncharted Territory. Routledge, 2014, 120–140.
40. DeMarco, T. Controlling Software Projects: Management, Measurement and Estimates. Prentice Hall, 1986, 30–68.
41. Duncan, W. A Guide to the Project Management Body of Knowledge. Project Management Institute, 1996, 1–45.
42. Dvir, D., & Lechler, T. Plans Are Nothing, Changing Plans Is Everything: The Impact of Changes on Project Success. Research Policy, 2004, 33(1), 1–15.
43. Eisenhardt, K. M. Agency Theory: An Assessment and Review. Academy of Management Review, 1989, 14(1), 57–74.
44. Fielding, R. T., Gettys, J., Mogul, J. C., et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, 1999.

45. Flyvbjerg, B. What You Should Know About Megaprojects and Why: An Overview. Project Management Journal, 2014, 45(2), 6–19.
46. Fowler, M., & Highsmith, J. The Agile Manifesto. Software Development, 2001, 9(8), 28–35.
47. Galliers, R. D., & Leidner, D. E. Strategic Information Management: Challenges and Strategies in Managing Information Systems. 3rd Edition, Routledge, 2003, 45–70.
48. Garcia, S., & You, X. Project Management Metrics, KPIs, and Dashboards. Auerbach Publications, 2015, 10–40.
49. Ghezzi, C., Jazayeri, M., & Mandrioli, D. Fundamentals of Software Engineering. 2nd Edition, Prentice Hall, 2003, 55–95.
50. Glorot, X., Bordes, A., & Bengio, Y. Deep Sparse Rectifier Neural Networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 2011, 15, 315–323.
51. Git (Torvalds, L., & Hamano, J.). Git: Fast Version Control System. [Online]. Available: https://git-scm.com
52. Gomede, E. Unraveling the Complexity: A Comparative Analysis of Neural Network Topology and Architecture for Project Planning. AImonks, 2023. [Online].Available:https://medium.com/aimonks/unraveling-the-complexity-a-comparative-analysis-of-neural-network-topology-and-architecture-for-15bd2973a622
53. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, 2016, 315–350.
54. Graves, A. Supervised Sequence Labelling with Recurrent Neural Networks. Springer, 2012, 12–25.
55. Harris, C. R., Millman, K. J., van der Walt, S. J., et al. Array Programming with NumPy. Nature, 2020, 585(7825), 357–362.

56. Hastie, T., Tibshirani, R., & Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd Edition, Springer, 2009, 389–400.
57. Hillson, D. Managing Risk in Projects. Routledge, 2017, 145–167.
58. Hinton, G. Neural Networks for Machine Learning. Coursera Lecture 6.5. [Online]. Available: https://www.coursera.org/learn/neural-networks, 2012.
59. Humphrey, W. S. Managing the Software Process. Addison-Wesley, 1989, 22–50.
60. Johansen, A. Project Uncertainty Management: A New Approach. International Journal of Project Management, 2015, 33(3), 24–29.
61. Jones, M., Bradley, J., & Sakimura, N. JSON Web Token (JWT). RFC 7519, 2015.
62. Jouppi, N. P., Young, C., Patil, N., et al. In-datacenter Performance Analysis of a Tensor Processing Unit. Proceedings of the 44th Annual International Symposium on Computer Architecture, 2017, 1–12.
63. Kaggle (n.d.). [Online]. Available: https://www.kaggle.com
64. Keil, M., Cule, P., Lyytinen, K., & Schmidt, R. A Framework for Identifying Software Project Risks. Communications of the ACM, 1998, 41(11), 76–83.
65. Kelley, J. E., & Walker, M. R. Critical-Path Planning and Scheduling. Proceedings of the Eastern Joint Computer Conference, 1959, 160–173.
66. Kerzner, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. John Wiley & Sons, 2017, 129–133.
67. Kim, S., & Kim, H. A New Metric of Absolute Percentage Error for Intermittent Demand Forecasts. International Journal of Forecasting, 2016, 32(3), 669–679.
68. Kotonya, G., & Sommerville, I. Requirements Engineering: Processes and Techniques. Wiley, 1998, 10–48.

69. Larman, C. Agile and Iterative Development: A Manager's Guide. Addison-Wesley, 2004, 215–220.
70. LeCun, Y., Bengio, Y., & Hinton, G. Deep Learning. Nature, 2015, 521(7553), 436–444.
71. Leach, P., Mealling, M., & Salz, R. A Universally Unique Identifier (UUID) URN Namespace. IETF RFC 4122, 2005.
72. Li, H., Lu, W., & Huang, T. Rethinking Project Management: A Focus on Knowledge Management. IEEE Transactions on Engineering Management, 2009, 56(4), 599–601.
73. Li, J., Zhao, Z., & Cai, H. Project Risk Prediction Using Integrated Machine Learning Approaches. Expert Systems with Applications, 2021, 176, 114825.
74. Liu, D. Project Portfolio Selection with Fuzzy Multi-Criteria Decision Making Methods. In Soft Computing in Management and Business Economics, 2018, 91–106.
75. Lu, H., Li, Y., & Chen, M. A Project Management Maturity Model for AI-Driven Organizations. International Journal of Information Management, 2020, 50, 476–489.
76. Ma, P., & Xin, J. A Hybrid Machine Learning Model for Software Project Cost Estimation. Applied Sciences, 2022, 12(1), 467.
77. MacCormack, A., & Herman, K. Agile Project Management in Large-Scale Systems. Harvard Business Review, 2019, 12(4), 88–94.
78. McCarthy, J. What Is Artificial Intelligence? Stanford University AI Archives, 2007, 1–15.
79. Mills, H. Software Development Project Management. IEEE Transactions on Software Engineering, 1983, (5), 265–270.
80. Mitchell, T. M. Machine Learning. McGraw-Hill, 1997, 100–145.

81. Molokken, K., & Jorgensen, M. A Review of Surveys on Software Effort Estimation. Proceedings of the 2003 International Symposium on Empirical Software Engineering, 2003, 223–230.
82. Montgomery, D. C., & Runger, G. C. Applied Statistics and Probability for Engineers. 7th Edition, John Wiley & Sons, 2018, 256–258.
83. Nielsen, M. A. Neural Networks and Deep Learning. Determination Press, 2015, 120–145.
84. North, K. Project and Risk Management: A Holistic Approach. Springer, 2011, 55–92.
85. NVIDIA (n.d.). CUDA Toolkit Documentation. [Online]. Available: https://developer.nvidia.com/cuda-toolkit
86. NVIDIA (n.d.). cuDNN: GPU-Accelerated Library for Deep Neural Networks. [Online]. Available: https://developer.nvidia.com/cudnn
87. Oh, C., & Lee, S. Hybrid Deep Learning-Based Risk Classification Approach for Project Management. Expert Systems with Applications, 2020, 147, 113202.
88. OpenAPI Initiative. OpenAPI Specification. GitHub repository. https://github.com/OAI/OpenAPI-Specification, 2021.
89. Papke-Shields, K. E., & Boyer-Wright, K. M. Strategic Planning Characteristics Applied to Project Management. International Journal of Project Management, 2017, 35(2), 169–179.
90. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011, 12, 2825–2830.
91. Phillips, J. PMP Project Management Professional Study Guide. 5th Edition, McGraw-Hill, 2010, 200–230.

92. PostgreSQL Global Development Group (n.d.). PostgreSQL Documentation: Index Types. [Online]. Available: https://www.postgresql.org/docs/current/indexes-types.html
93. Powers, D. M. Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies, 2011, 2(1), 37–63.
94. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide) – 7th Edition. PMI, 2021, 10–80.
95. Pyzdek, T., & Keller, P. The Six Sigma Handbook. McGraw-Hill Education, 2018, 50–95.
96. Rapp, K. Mastering JIRA 7: An Expert Guide to Building and Managing JIRA Projects. Packt Publishing, 2018, 89–105.
97. Raz, T., Shenhar, A. J., & Dvir, D. Risk management, project success, and technological uncertainty. R&D Management, 2002, 101–109.
98. Rehman, F. A Survey on Machine Learning in Project Management. International Journal of Advanced Computer Science and Applications, 2019, 10(5), 147–158.
99. Ropponen, J., & Lyytinen, K. Components of Software Development Risk: How to Address Them? A Project Manager Survey. IEEE Transactions on Software Engineering, 2000, 26(2), 98–112.
100. Roy, S. Machine Learning for Software Project Effort Estimation: Review and Open Challenges. ACM Computing Surveys, 2021, 54(2), 1–36.
101. Russell, S., & Norvig, P. Artificial Intelligence: A Modern Approach. 4th Edition, Pearson, 2020, 450–460.
102. Schmidt, R. Decision Support Systems in Project Management. Springer, 2017, 10–50.
103. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. Neural Networks, 2015, 61, 85–117.

104. Shoorkand, H. D., Nourelfath, M., & Hajji, A. A Hybrid Deep Learning Approach to Integrate Predictive Maintenance Planning. Journal of Manufacturing Systems, 2024, 397–410. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612524000712
105. Singh, G. Machine Learning in Agile Project Management: A Systematic Literature Review. Journal of Software, 2020, 15(8), 345–352.
106. Smith, R. Data-Driven Project Management. Wiley, 2015, 60–85.
107. Sommerville, I. Software Engineering. 10th Edition, Pearson, 2015, 70–86.
108. Sommerville, I., & Sawyer, P. Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997, 45–88.
109. Spring Boot (Walls, C.). Spring Boot in Action. Manning Publications, 2016.
110. Sutherland, J., & Schwaber, K. The Scrum Guide. Scrum.org, 2016, 1–19.
111. Syed, A. Fuzzy Logic Approaches to Risk Assessment in Software Projects. Information and Software Technology, 2021, 137, 106587.
112. Taylor, J. A Survival Guide for Project Managers. AMACOM, 2006, 45–79.
113. Thamhain, H. Managing Technology-Based Projects: Tools, Techniques, People and Business Processes. John Wiley & Sons, 2014, 55–90.
114. Turner, J. R. Handbook of Project-Based Management. McGraw-Hill, 2014, 1–50.
115. Tyrväinen, O., & Peltonen, J. Real-Time Estimation of Backlog Item Durations with Neural Networks: A JIRA-Based Approach. Journal of Systems and Software, 2019, 158, 110426.
116. Ulrich, W., & Eppinger, S. Product Design and Development. McGraw-Hill, 2011, 40–69.

117. Vaida, G., & Kumar, V. A Machine Learning-Driven Approach for Task Prioritization in Agile Software Development. Journal of Information Technology Management, 2022, 33(2), 121–130.
118. Varajão, J., & Trigo, A. Improving Project Management Practice: A Collaborative Approach. Business Process Management Journal, 2016, 22(5), 924–941.
119. Walls, C. Spring Boot in Action. Manning Publications, 2016.
120. Wang, J. Risk-Aware Scheduling in Project Management: A Machine Learning Approach. International Journal of Production Economics, 2020, 231, 107917.
121. White, D., & Fortune, J. Current Practice in Project Management — An Empirical Study. International Journal of Project Management, 2002, 20(1), 1–11.
122. Williams, R. Agile Metrics in Action: Measuring and Enhancing the Performance of Agile Teams. Manning Publications, 2022, 50–90.
123. Williams, T. M. Assessing and Moving on From the Dominant Project Management Discourse in the Light of Project Overruns. IEEE Transactions on Engineering Management, 2003, 497–508.
124. Wilson, P. The Project Management Book: How to Run Successful Projects in Half the Time. Kogan Page, 2016, 60–95.
125. Xia, W., & Lee, G. Grasping the Complexity of IS Development Projects. Communications of the ACM, 2004, 47(5), 68–74.
126. Xu, Q., & Zhao, H. Predicting Software Development Effort Using Ensemble Learning. Journal of Systems and Software, 2019, 149, 43–54.
127. Yang, L., & Cai, J. A Fuzzy Decision-Making Approach for Agile Project Management. Soft Computing, 2020, 24(5), 3573–3582.
128. Young, T. L. Successful Project Management. Kogan Page, 2016, 22–66.

129. Yu, H., & Chen, M. Risk-Based Scheduling Using Support Vector Machines. Computers & Industrial Engineering, 2019, 127, 743–750.
130. Zhang, B. Intelligent Automation in Project Management. IEEE Transactions on Engineering Management, 2020, 67(4), 1082–1092.
131. Zhang, X., & Fan, Z. Improving the Performance of Project Management by Using Machine Learning Techniques. Expert Systems with Applications, 2010, 37(6), 7592–7596.
132. Zhang, Y., & Patel, S. Reinforcement Learning for Agile Project Scheduling. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33, 9751–9752.
133. Zhou, K., Fu, C., & Yang, S. Intelligent Project Scheduling Using Deep Reinforcement Learning. Applied Soft Computing, 2021, 113, 107969.
134. Zhou, M., & Chen, W. Project Scheduling under Uncertain Conditions. Computers & Industrial Engineering, 2018, 123, 19–28.
135. Zhou, M. Big Data Analytics in Agile Project Management: A Case Study. Information & Management, 2019, 56(5), 665–672.
136. Zhou, R. Investigating Machine Learning-Based Risk Management in Agile Environments. Journal of Systems and Software, 2020, 168, 110609.
137. Zhou, X. Machine Learning-Based Agile Resource Optimization. Expert Systems with Applications, 2021, 185, 115623.
138. Zin, A. Knowledge-Based Risk Classification for Software Projects. International Journal of Information Management, 2019, 47, 151–159.
139. Zimmer, B. Automatic Detection of Project Bottlenecks Using Text Mining. Information Processing & Management, 2021, 58(4), 102590.
140. Zimmerman, J. Data-Driven Project Governance: A Machine Learning Approach. Information & Management, 2020, 57(2), 103180.

141. Zubo, R., & Jacobs, T. Resource Optimization in Multi-Project Environment: A Machine Learning Approach. Computers & Operations Research, 2018, 92, 35–47.

142. Zuo, Y. Neural Net-Based Approach for Project Task Prioritization. Knowledge-Based Systems, 2018, 159, 191–199.

143. Zyb, P. ML-Driven Cost Forecasting in Agile Software Projects. Software Quality Journal, 2020, 28(4), 763–780.

144. Zymo, H. A Reinforcement Learning Approach for Stakeholder Analysis in Large Projects. Expert Systems with Applications, 2022, 189, 116083.