Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

До захисту допущено
Кафедрою математичного моделювання та аналізу даних
протокол №___ від _____

Завідувач кафедри _____ Володимир СТРУКОВ
(підпис) (імя, прізвище)

«___»_____ 2026 р.

Кваліфікаційна робота
здобувача першого (бакалаврського) рівня вищої освіти

АНАЛІЗ ВПЛИВУ AI-ІНСТРУМЕНТІВ НА РОЛЬ PROJECT ТА PRODUCT MANAGER

Спеціальність (спеціалізація) 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконавець _____ Бірюкова Ніка
(підпис) (ім'я, прізвище)

Науковий керівник _____ Лисицький Костянтин
(підпис) (ім'я, прізвище)

Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Кафедра математичного моделювання та аналізу даних

ЗАТВЕРДЖУЮ
Завідувач кафедри ММтАД

_____Володимир СТРУКОВ
підпис ім'я, прізвище

"____" _____2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувача першого (бакалаврського) рівня вищої освіти

Бірюкова Ніка Сергіївна

Спеціальність (спеціалізація) 122 «Комп'ютерні науки», _____

Освітня програма Комп'ютерні системи та мережі _____

1. Тема роботи: АНАЛІЗ ВПЛИВУ AI-ІНСТРУМЕНТІВ НА РОЛЬ PROJECT TA PRODUCT MANAGER

керівник роботи ЛИСИЦЬКИЙ К.Є., доцент кафедри математичного моделювання та аналізу даних

затверджені наказом по Університету від 29 січня 2026 року № 4101-5/225

2. Строк здобувачем роботи "____" _____2026 року

3. Вихідні дані до роботи: структурований датасет параметрів IT-завдань (1000 записів); ансамблеві алгоритми машинного навчання (Random Forest) та методи NLP; мова Python із застосуванням бібліотеки Scikit-learn і фреймворку Streamlit.

4. Перелік питань, які потрібно розробити:

1. Провести аналіз предметної області та обґрунтувати актуальність автоматизації управління Agile-проєктами.

2. Дослідити математичні методи обробки природної мови (NLP) та алгоритми машинного навчання для прогнозування параметрів беклогу.
3. Спроєктувати гібридну архітектуру програмного прототипу AI-асистента (семантичний, предиктивний та аналітичний модулі).
4. Підготувати навчальний набір даних (датасет) та виконати розвідувальний аналіз даних (EDA).
5. Здійснити програмну реалізацію системи (навчання моделей Random Forest, створення інтерактивного вебзастосунку).
6. Виконати тестування розробленого рішення та оцінити метрики якості предиктивних моделей.

5. План роботи

№ з/п	Назви етапів роботи	Строк виконання етапів роботи	Примітка
1	Формулювання мети та повного переліку завдань для виконання кваліфікаційної роботи.	11.01.2026 – 24.01.2026	Виконано
2	Пошук та аналіз інформаційних джерел за темою КР.	24.01.2026 – 15.02.2026	Виконано
3	Проєктування архітектури гібридного AI-асистента	15.02.2026 – 13.03.2026	Виконано
4	Формування навчального датасету та програмна реалізація системи	13.03.2026 – 28.04.2026	Виконано
5	Тестування програмного прототипу, оцінювання метрик якості моделей та розвідувальний аналіз даних (EDA).	28.04.2026 – 14.04.2026	Виконано
6	Підсумок отриманих результатів та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.	14.04.2026 – 01.05.2026	Виконано

6. Дата видачі завдання 11 січня 2026 р.

Здобувач вищої освіти _____ Бірюкова Н.С.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Лисицький К.Є.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота присвячена аналізу впливу AI-інструментів на управління програмними проєктами. Мета — дослідження можливостей ШІ для автоматизації управлінських задач та розробка прототипу AI-асистента для Project і Product менеджерів.

Об’єкт дослідження — процеси управління проєктами в межах життєвого циклу розробки ПЗ. Предмет дослідження — методи ШІ для автоматизації рутини, оцінювання складності та прогнозування ризиків.

Методи дослідження включають системний аналіз, обробку природної мови (NLP), машинне навчання та експериментальне тестування. Практична частина реалізована мовою Python із використанням бібліотек Pandas, NumPy, Scikit-learn, Joblib, Streamlit та Matplotlib.

У результаті розроблено програмний прототип AI-асистента, який дозволяє формувати структуровану Jira-задачу, User Story, Acceptance Criteria та визначати потенційні ризики. На базі синтетичного датасету (1000 задач) навчено моделі: RandomForestRegressor та RandomForestClassifier.

Новизна роботи полягає у запропонованому гібридному підході, який поєднує Semantic Engine, Predictive Engine та Data Analytics Module. Результати підтверджують доцільність використання AI/ML як інструменту для автоматизації задач, попередньої оцінки складності та підтримки планування спринтів.

Ключові слова: штучний інтелект, машинне навчання, Project Manager, Product Manager, NLP, AI-асистент, Story Points, прогнозування ризиків, Streamlit, Random Forest, управління проєктами.

ABSTRACT

This qualification paper is dedicated to analyzing the impact of AI tools on software project management. The aim is to explore the capabilities of AI in automating managerial tasks and to develop a prototype of an AI assistant for Project and Product Managers.

The object of research is the project management processes within the software development life cycle (SDLC). The subject of research encompasses AI methods used for automating routine tasks, estimating complexity, and predicting risks.

Research methods include system analysis, natural language processing (NLP), machine learning, and experimental testing. The practical part is implemented in Python using Pandas, NumPy, Scikit-learn, Joblib, Streamlit, and Matplotlib libraries.

As a result, a software prototype of an AI assistant was developed. It allows for the generation of structured Jira tasks, User Stories, and Acceptance Criteria, as well as the identification of potential risks. Based on a synthetic dataset (1,000 tasks), models were trained: RandomForestRegressor and RandomForestClassifier.

The novelty of the work lies in the proposed hybrid approach, which combines a Semantic Engine, a Predictive Engine, and a Data Analytics Module. The results confirm the feasibility of using an AI/ML approach as a tool for automating tasks, conducting preliminary complexity estimations, and supporting sprint planning.

Keywords: artificial intelligence, machine learning, Project Manager, Product Manager, NLP, AI assistant, Story Points, risk prediction, Streamlit, Random Forest, project management

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Accuracy	-	Частка правильно класифікованих об’єктів
AI	-	Artificial Intelligence, штучний інтелект
API	-	Application Programming Interface, програмний інтерфейс взаємодії
CSV	-	Формат збереження табличних даних
F1-score	-	Узагальнена метрика, що враховує Precision і Recall
LLM	-	Large Language Model, велика мовна модель
MAE	-	Mean Absolute Error, середня абсолютна помилка
ML	-	Machine Learning, машинне навчання
MSE	-	Mean Squared Error, середньоквадратична помилка
NLP	-	Natural Language Processing, обробка природної мови
PM	-	Project Manager, менеджер проєкту
Precision	-	Точність позитивних прогнозів певного класу
R2 Score	-	Коефіцієнт детермінації регресійної моделі
Recall	-	Повнота виявлення об’єктів певного класу
RMSE	-	Root Mean Squared Error, корінь із середньоквадратичної помилки
SDLC	-	Software Development Life Cycle, життєвий цикл розробки програмного забезпечення
UI	-	User Interface, користувацький інтерфейс
UX	-	User Experience, користувацький досвід

ВСТУП

У сучасних умовах цифрової трансформації управління програмними проєктами зазнає суттєвих змін. Зростання складності програмних продуктів, збільшення обсягів даних, постійна зміна вимог користувачів і скорочення строків розробки зумовлюють потребу у використанні інтелектуальних інструментів підтримки діяльності IT-команд. Особливо актуальним це є для ролей Project Manager та Product Manager, які відповідають за планування, координацію, пріоритизацію задач, формування вимог, управління ризиками та забезпечення цінності програмного продукту.

Управління програмними проєктами охоплює планування, організацію роботи команди, контроль строків, оцінювання ресурсів, управління ризиками та забезпечення досягнення цілей проєкту. У сучасних IT-компаніях ці процеси часто реалізуються в межах Agile-підходів, зокрема Scrum і Kanban, що дозволяють швидко реагувати на зміни вимог і поступово вдосконалювати програмний продукт [10-12]. Project Manager відповідає за організацію процесу розробки, контроль строків, розподіл ресурсів і зниження ризиків проєкту. Product Manager зосереджується на формуванні бачення продукту, аналізі потреб користувачів, управлінні беклогом і визначенні пріоритетів [9, 14].

Сучасний стан предметної області характеризується активним впровадженням AI-інструментів у процеси управління програмними проєктами. Штучний інтелект використовується для генерації текстової інформації, автоматизації підготовки документації, аналізу задач, прогнозування ризиків і підтримки прийняття рішень [1-3]. Такі інструменти, як Jira AI, Notion AI, Linear, ClickUp AI, ChatGPT і GitHub Copilot, уже застосовуються для автоматизації окремих етапів роботи з вимогами, задачами, документацією та програмним кодом.

Водночас більшість наявних рішень орієнтована або на генерацію тексту, або на автоматизацію окремих дій у межах конкретної платформи. Для діяльності Project та Product Manager важливим є комплексний підхід, який дозволяє одночасно працювати з текстовими описами задач, числовими параметрами проєкту, прогнозуванням ризиків і візуальною аналітикою. Саме тому актуальною є задача розробки програмного прототипу, що поєднує кілька AI/ML-компонентів в єдиній системі підтримки управлінських рішень.

Актуальність теми також зумовлена тим, що значна частина діяльності Project та Product Manager має рутинний характер. До таких задач належать створення описів задач, формування User Story, написання Acceptance Criteria, попереднє оцінювання складності, аналіз ризиків, перевірка беклогу й контроль виконання спринту. Виконання цих операцій вручну потребує значних часових витрат і може призводити до суб’єктивності оцінок або несвоєчасного виявлення ризиків. У таких умовах AI-інструменти можуть виступати не заміною менеджера, а допоміжним інструментом, який підвищує швидкість обробки інформації та надає додаткову аналітичну підтримку.

Особливо важливою є задача прогнозування складності, тривалості виконання та ризику задач. У реальних програмних проєктах ці параметри залежать від багатьох факторів: пріоритету задачі, її типу, складності, кількості залежностей, розміру команди, досвіду розробника, дня спринту, потреби у зовнішній інтеграції, тестуванні або дизайні. Традиційне оцінювання таких параметрів часто базується на експертному досвіді команди, що може бути суб’єктивним. Використання моделей машинного навчання дозволяє доповнити експертні оцінки даними та виявляти закономірності, які не завжди очевидні під час ручного аналізу [18, 21, 22].

Метою дослідження є аналіз впливу AI-інструментів на діяльність Project та Product Manager, а також розробка програмного прототипу AI-асистента, який поєднує методи обробки природної мови, машинного

навчання та візуальної аналітики для автоматизації формування задач, оцінювання трудомісткості та прогнозування ризиків у програмних проєктах.

Об’єктом дослідження є процеси управління програмними проєктами та діяльність Project і Product Manager у межах життєвого циклу розробки програмного забезпечення.

Предметом дослідження є методи та програмні засоби штучного інтелекту, зокрема NLP-підходи, регресійні моделі машинного навчання, класифікаційні моделі та засоби візуальної аналітики, що застосовуються для автоматизації управлінських задач, оцінювання складності та прогнозування ризиків у програмних проєктах.

Для досягнення поставленої мети необхідно виконати такі завдання: проаналізувати роль Project Manager та Product Manager у процесах розробки програмного забезпечення; дослідити сучасні AI-інструменти та методи штучного інтелекту для автоматизації управлінських задач; обґрунтувати доцільність гібридного підходу; розробити архітектуру програмного прототипу; сформувати датасет задач програмного проєкту; реалізувати моделі прогнозування Story Points, тривалості виконання та рівня ризику; створити користувацький інтерфейс і провести тестування системи.

У роботі використано методи системного аналізу, аналізу наукових джерел, обробки природної мови, машинного навчання, регресійного аналізу, класифікації, візуального аналізу даних та експериментального тестування. Вони застосовуються для формалізації задач Project та Product Management, побудови моделей прогнозування, аналізу результатів і перевірки працездатності програмного прототипу.

Науково-практична новизна одержаних результатів полягає у запропонованому підході до підтримки діяльності Project та Product Manager шляхом інтеграції семантичної обробки опису задачі, генерації структурованої управлінської документації та ML-прогнозування параметрів задачі в одному

програмному прототипі. На відміну від підходів, які орієнтовані лише на генерацію тексту або лише на аналітичне прогнозування, у роботі запропоновано гібридну архітектуру AI-асистента, що поєднує Semantic Engine, Predictive Engine та Data Analytics Module. Це дозволяє не лише формувати текстовий опис задачі, User Story та Acceptance Criteria, а й прогнозувати Story Points, тривалість виконання, рівень ризику та ймовірність затримки.

Практичне значення одержаних результатів полягає у розробці програмного прототипу AI-асистента, який може бути використаний як допоміжний інструмент для формування задач, створення User Story та Acceptance Criteria, попереднього оцінювання складності задач, прогнозування строків виконання й аналізу ризиків. Розроблена система дозволяє скоротити час на виконання рутинних управлінських операцій, підвищити структурованість задач у беклозі та надати Project або Product Manager додаткову аналітичну інформацію для планування спринту.

Практична частина роботи реалізована мовою програмування Python із використанням бібліотек Pandas, NumPy, Scikit-learn, Joblib, Streamlit і Matplotlib. Для прогнозування Story Points та тривалості виконання задач використано моделі RandomForestRegressor, а для визначення рівня ризику — RandomForestClassifier. Користувацький інтерфейс реалізовано у вигляді Streamlit-застосунку.

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

1. ТЕОРЕТИЧНІ ОСНОВИ ВИКОРИСТАННЯ AI-ІНСТРУМЕНТІВ У PROJECT ТА PRODUCT MANAGEMENT

1.1. Постановка задачі дослідження та роль Project і Product Manager у життєвому циклі розробки програмного забезпечення

У сучасних умовах цифрової трансформації управління програмними проєктами стає складним процесом, що потребує швидкого прийняття рішень, координації між командами, контролю строків, ризиків і якості розробки. Особливо актуальним це є для IT-компаній, де програмні продукти створюються в умовах змінних вимог, обмежених ресурсів і високої конкуренції. У таких умовах важливу роль відіграють Project Manager та Product Manager, які забезпечують організацію процесу розробки, планування робіт, комунікацію із зацікавленими сторонами та досягнення бізнес-цілей продукту [10-12].

Project Manager відповідає переважно за організаційний і процесний бік розробки програмного забезпечення. Його діяльність пов’язана з плануванням задач, розподілом ресурсів, контролем строків, управлінням ризиками та координацією роботи команди. У сфері IT-проєктів важливе значення мають гнучкі методології управління, зокрема Agile, Scrum і Kanban, які дозволяють швидко реагувати на зміну вимог і забезпечувати поступову розробку програмного продукту [10, 13, 20].

Product Manager, на відміну від Project Manager, більше зосереджується на цінності продукту для користувача та бізнесу. Його основні функції пов’язані з формуванням бачення продукту, управлінням беклогом, визначенням пріоритетів, аналізом потреб користувачів і перевіркою відповідності продукту очікуванням ринку. У сучасних умовах цифровізації роль Product Manager поступово трансформується, оскільки він дедалі частіше працює з аналітичними даними, AI-інструментами та системами підтримки прийняття рішень [9].

У межах життєвого циклу розробки програмного забезпечення Project Manager та Product Manager взаємодіють на різних етапах: від аналізу ідеї та формування вимог до планування спринтів, контролю виконання задач, тестування, релізу та подальшого розвитку продукту. Їхня діяльність охоплює як стратегічні, так і рутинні задачі. До стратегічних належать визначення цілей продукту, планування розвитку, управління ризиками та пріоритетами. До рутинних — створення описів задач, формування User Story, Acceptance Criteria, оцінювання складності, підготовка документації та аналіз беклогу.

Саме рутинні та повторювані задачі є найбільш перспективними для автоматизації за допомогою штучного інтелекту. AI-система може допомогти менеджеру швидше формувати опис задачі, створювати критерії приймання, попередньо оцінювати складність задачі, прогнозувати її тривалість і виявляти потенційні ризики. Це не означає повну заміну Project або Product Manager, а радше зміну їхньої ролі: менеджер переходить від ручного виконання однотипних операцій до контролю якості AI-рішень, інтерпретації результатів і прийняття управлінських рішень [1, 2, 19].

Отже, проблема дослідження полягає в необхідності підвищення ефективності роботи Project та Product Manager за рахунок інструментів, які можуть одночасно обробляти текстову та числову інформацію про задачі. У межах цієї роботи розглядається підхід, за якого короткий опис задачі перетворюється на структуровану Jira-задачу, доповнюється User Story та Acceptance Criteria, а також використовується для прогнозування трудомісткості, тривалості й ризику затримки. Такий підхід створює основу для розробки програмного прототипу AI-асистента, що поєднує текстову генерацію, машинне навчання та візуальну аналітику.

1.2. Аналіз сучасних AI-інструментів для підтримки управління програмними проєктами

Останніми роками штучний інтелект активно впроваджується у сферу управління проєктами. Це пов’язано з розвитком великих мовних моделей, генеративного AI, систем автоматизації робочих процесів та інструментів аналізу даних. У наукових дослідженнях зазначається, що AI може використовуватися для прогнозування ризиків, автоматизації планування, аналізу продуктивності команд, підтримки прийняття рішень і формування проєктної документації [1, 3, 21, 22].

Серед сучасних інструментів, які застосовуються у Project та Product Management, можна виділити Jira, Notion, Linear, ClickUp, GitHub Copilot, ChatGPT та інші AI-рішення. Jira використовується для управління задачами, спринтами й беклогом, Notion — для організації документації та командної роботи, Linear — для роботи продуктових та інженерних команд із задачами й релізами, а GitHub Copilot переважно підтримує розробників у написанні коду. ChatGPT та подібні інструменти можуть допомагати з генерацією текстів, поясненням вимог, підготовкою документації та структуруванням ідей.

AI-функції в таких системах можуть автоматизувати частину рутинної роботи менеджера: формувати чорнові описи задач, створювати User Story та Acceptance Criteria, підсумовувати інформацію, аналізувати беклог, допомагати з пріоритизацією або формувати рекомендації. Це особливо корисно для Agile-команд, де задачі постійно оновлюються, уточнюються та проходять через різні статуси [2, 5, 14].

Водночас сучасні AI-рішення мають певні обмеження. Багато з них орієнтовані переважно на генерацію тексту, але не завжди виконують кількісне прогнозування параметрів задач, таких як Story Points, тривалість виконання або ризик затримки. Крім того, частина інструментів працює всередині конкретної платформи й не завжди може бути гнучко адаптована до

потреб окремої команди. Також результати AI-інструментів потребують перевірки з боку менеджера, оскільки можуть бути неточними, надто загальними або недостатньо прив’язаними до контексту конкретного проєкту [19, 24].

Окрему роль у Project Management відіграє управління ризиками. Ризики можуть бути пов’язані з технічною складністю задачі, зовнішніми інтеграціями, нестачею ресурсів, залежностями між командами, зміною вимог або недостатньою деталізацією задачі. У наукових працях підкреслюється, що ризик-орієнтований підхід є важливою складовою ефективного управління IT-проєктами, оскільки дозволяє заздалегідь виявляти потенційні проблеми та зменшувати ймовірність зриву строків [15, 16, 17].

AI-інструменти можуть бути корисними для управління ризиками, оскільки здатні аналізувати історичні дані, виявляти закономірності та прогнозувати проблемні задачі. Для Product Manager такі інструменти також можуть допомагати з аналізом беклогу, формуванням пріоритетів, підготовкою вимог і виявленням дублювання задач. Це дозволяє пришвидшити обробку великої кількості ідей, звернень користувачів або внутрішніх запитів команди [9, 14].

Порівняльний аналіз сучасних AI-рішень показує, що більшість із них вирішує окремі задачі: генерацію тексту, автоматизацію нотаток, підсумовування, створення задач або підтримку програмування. Проте для діяльності Project та Product Manager важливим є комплексний підхід, який поєднує текстову обробку, прогнозування та аналітику. Саме тому в межах цієї роботи доцільно розглядати не окремий AI-інструмент, а гібридну систему, яка об’єднує кілька функціональних модулів.

Таким чином, сучасні AI-рішення створюють основу для автоматизації управлінських процесів у програмних проєктах, але залишають простір для вдосконалення. Основним напрямом такого вдосконалення є поєднання

можливостей генеративного AI, машинного навчання та аналітики даних в одному програмному прототипі, орієнтованому на практичні потреби Project та Product Manager.

1.3. Аналіз методів штучного інтелекту для автоматизації управлінських задач та обґрунтування гібридного підходу

Для автоматизації задач Project та Product Manager можуть використовуватися різні методи штучного інтелекту. До них належать обробка природної мови, великі мовні моделі, векторні представлення тексту, методи розпізнавання сутностей, регресійні моделі, класифікаційні алгоритми, дерева рішень, ансамблеві моделі та нейронні мережі. Кожен із цих підходів має власні переваги, обмеження та сферу застосування [7, 8, 18, 24].

Обробка природної мови, або NLP, використовується для аналізу, інтерпретації та генерації текстової інформації. У контексті управління проєктами NLP може застосовуватися для обробки описів задач, виділення ключових понять, класифікації запитів, формування User Story, Acceptance Criteria або коротких резюме. Перевагою NLP є здатність працювати з неструктурованими текстовими даними, які часто трапляються в беклогах, описах задач, коментарях і документації. Недоліком є залежність результату від якості вхідного тексту, контексту задачі та можливостей обраної моделі [7, 8].

Великі мовні моделі, або LLM, є розвитком NLP-підходів і дозволяють генерувати змістовні тексти, відповідати на запитання, структурувати інформацію та виконувати роль інтелектуального помічника. Для Product Manager такі моделі корисні під час формування вимог, написання User Story або підготовки описів функціональності. Для Project Manager LLM можуть допомагати створювати звіти, підсумовувати статуси задач і формувати рекомендації. Водночас LLM можуть генерувати правдоподібні, але неточні

відповіді, тому результати їхньої роботи потребують контролю з боку людини [1, 7, 21].

Embeddings, або векторні представлення тексту, дозволяють перетворювати текстову інформацію у числовий вигляд. Завдяки цьому можна порівнювати описи задач, знаходити схожі або дубльовані елементи беклогу, групувати задачі за тематикою та виконувати семантичний пошук. NER, або виділення іменованих сутностей, може застосовуватися для автоматичного знаходження у тексті назв модулів, ролей користувачів, компонентів системи, технологій або строків. У задачах Project та Product Management ці методи допомагають структурувати неформальні описи ідей і перетворювати їх на формалізовані задачі.

Для кількісного прогнозування в управлінні проєктами застосовуються методи машинного навчання. Регресійні моделі використовуються тоді, коли потрібно передбачити числове значення, наприклад Story Points або тривалість виконання задачі в годинах. Класифікаційні моделі застосовуються для визначення категорії, наприклад рівня ризику: низький, середній або високий. Такі моделі можуть враховувати різні ознаки задачі: складність, пріоритет, кількість залежностей, розмір команди, день спринту, досвід розробника, потребу в тестуванні або зовнішній інтеграції [18, 21, 22].

Серед алгоритмів машинного навчання для цієї роботи доцільним є використання Random Forest. Цей метод належить до ансамблевих алгоритмів і поєднує багато дерев рішень, що дозволяє отримати стабільніший прогноз порівняно з одним деревом. Random Forest добре працює з табличними даними, підтримує як регресію, так і класифікацію та не потребує надмірно складної підготовки даних. XGBoost і нейронні мережі також можуть застосовуватися для подібних задач, однак вони потребують ретельнішого налаштування або більшого обсягу даних. Для бакалаврської роботи важливо

не лише отримати результат, а й пояснити логіку роботи системи, тому Random Forest є практичним і зрозумілим вибором [18].

Порівняльні переваги та недоліки основних методів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика методів штучного інтелекту для автоматизації управлінських задач

Метод	**Переваги**	**Недоліки**	**Доцільність у роботі**
NLP	Робота з текстовими описами задач	Залежність від якості тексту	Використовується для формування структурованої задачі
LLM	Генерація User Story, Acceptance Criteria, описів	Можливі неточності та потреба в контролі	Доцільно як Semantic Engine
Embeddings	Пошук схожих задач і дублікатів	Потребує додаткової моделі векторизації	Можливий напрям розвитку системи
Регресія	Прогноз числових показників	Помилка залежить від якості ознак	Використовується для Story Points і тривалості
Класифікація	Визначення рівня ризику	Може залежати від балансу класів	Використовується для Low / Medium / High risk
Random Forest	Стабільність, робота з табличними даними	Менша інтерпретованість, ніж у одного дерева	Обрано для практичної реалізації
Нейронні мережі	Можливість моделювати складні залежності	Потребують багато даних і ресурсів	Не є основним методом у цій роботі

У межах цієї дипломної роботи обрано гібридний підхід, який поєднує текстову обробку та машинне навчання. Такий підхід є доцільним, оскільки задачі Project та Product Manager не можна повністю звести лише до генерації тексту або лише до прогнозування чисел. Менеджеру потрібно одночасно отримати структурований опис задачі, оцінку її складності, прогноз строків, рівень ризику та рекомендацію щодо подальших дій.

Запропонований підхід складається з трьох основних частин. Перша частина — Semantic Engine, який відповідає за формування назви задачі, опису, User Story, Acceptance Criteria та ризиків. Друга частина — Predictive Engine, який використовує моделі машинного навчання для прогнозування Story Points, тривалості виконання та рівня ризику. Третя частина — Data Analytics Module, який забезпечує візуальний аналіз датасету, розподілу задач, ризиків, тривалості та кореляцій між ознаками.

Науково-практична новизна роботи полягає не в тому, що AI вперше застосовується в управлінні проєктами, а в інтеграції кількох підходів в одному програмному прототипі. Більшість існуючих AI-інструментів зосереджується або на генерації тексту, або на автоматизації окремих дій у системах управління задачами. У цій роботі запропоновано поєднати генерацію управлінської документації, прогнозування трудомісткості, оцінювання ризику та візуальну аналітику в єдиній системі підтримки Project та Product Manager.

Таким чином, гібридний підхід є найбільш доцільним для поставленої задачі, оскільки дозволяє враховувати як текстові, так і числові характеристики задачі. Це створює основу для більш обґрунтованого планування спринтів, оцінювання ризиків і підтримки прийняття рішень у процесі управління програмними проєктами.

Висновки до розділу 1

У першому розділі було розглянуто теоретичні основи використання AI-інструментів у діяльності Project та Product Manager. Встановлено, що сучасне управління програмними проєктами потребує швидкої обробки великої кількості інформації, а значна частина роботи менеджерів має рутинний і повторюваний характер.

Проаналізовані сучасні AI-інструменти здатні допомагати у створенні документації, формуванні задач і аналізі беклогу, проте вони часто вирішують лише окремі задачі й не завжди поєднують текстову генерацію, прогнозну аналітику та візуальний аналіз даних.

Розглянуто основні методи штучного інтелекту, які можуть застосовуватися для автоматизації управлінських задач (NLP, LLM, embeddings, NER, регресійні та класифікаційні моделі, Random Forest, XGBoost і нейронні мережі), та визначено їх переваги й недоліки. Обґрунтовано, що для практичної частини доцільно використати гібридний підхід, який поєднує Semantic Engine для генерації структурованої задачі, Predictive Engine для прогнозування параметрів та Data Analytics Module для аналізу даних.

Отже, результати теоретичного аналізу підтверджують доцільність розробки програмного прототипу AI-асистента. Така система може скоротити час на формалізацію задач, допомогти в оцінюванні складності, спрогнозувати ризики та надати менеджеру додаткову інформацію для планування спринту й прийняття управлінських рішень.

2. МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОТОТИПУ AI-АСИСТЕНТА

2.1. Концепція інтелектуальної системи та архітектура програмного рішення

На основі проведеного в першому розділі аналізу було визначено, що діяльність Project Manager та Product Manager у програмних проєктах містить значну кількість рутинних операцій, які можуть бути частково автоматизовані за допомогою AI-інструментів. До таких операцій належать формування описів задач, підготовка User Story, створення Acceptance Criteria, попереднє оцінювання складності задач, аналіз ризиків і підготовка рекомендацій для планування спринту. У зв’язку з цим у межах дипломної роботи було розроблено концепцію програмного прототипу AI-асистента, який поєднує можливості текстової генерації, машинного навчання та візуальної аналітики.

Основна ідея розробленої системи полягає в тому, щоб надати Project Manager або Product Manager інструмент для швидкого перетворення короткої ідеї чи неформального опису задачі у структурований формат, придатний для подальшого використання в системах управління задачами, наприклад Jira. Крім текстової частини, система виконує прогнозування ключових параметрів задачі: Story Points, орієнтовної тривалості виконання, рівня ризику та ймовірності затримки. Таким чином, програмний прототип виконує не лише функцію генератора тексту, а й функцію інструмента підтримки прийняття управлінських рішень.

Розроблена система отримала умовну назву AI-асистент для Project та Product Management. Вона орієнтована на підтримку діяльності менеджерів у процесах планування, деталізації та оцінювання задач у програмних проєктах. На відміну від традиційного підходу, коли менеджер вручну формує опис задачі, оцінює її складність і визначає ризики, запропонований прототип автоматизує частину цих дій. Це дозволяє скоротити час на підготовку задачі,

підвищити структурованість вимог і забезпечити попередню аналітичну оцінку ще до початку виконання задачі.

Концепція системи базується на гібридному підході, який передбачає поєднання декількох функціональних компонентів. У межах роботи виділено три основні модулі: Semantic Engine, Predictive Engine та Data Analytics Module. Semantic Engine відповідає за семантичну обробку короткого опису задачі та формування текстової частини результату. Predictive Engine виконує прогнозування параметрів задачі за допомогою моделей машинного навчання. Data Analytics Module забезпечує візуальний аналіз датасету та результатів роботи системи. Додатково система має користувацький інтерфейс, реалізований за допомогою Streamlit, який дозволяє вводити опис задачі, змінювати параметри, запускати прогнозування та переглядати результати.

Semantic Engine у розробленому прототипі виконує автоматичне визначення типу задачі, компонента системи та потреби в зовнішній інтеграції. Також цей модуль формує назву задачі, опис, User Story, Acceptance Criteria, список потенційних ризиків і рекомендацію для менеджера. У практичній реалізації застосовано спрощений NLP-підхід, заснований на аналізі ключових слів у тексті. Такий підхід є достатнім для дипломного прототипу, оскільки дозволяє продемонструвати базову логіку семантичного аналізу задачі без використання платних API або складних мовних моделей. У реальному впровадженні цей модуль може бути розширений за допомогою великих мовних моделей, embeddings або NER-механізмів, що дозволило б глибше аналізувати зміст задач і враховувати складніші мовні конструкції [7, 8, 24].

Predictive Engine є основним аналітичним модулем системи. Він використовує моделі машинного навчання для прогнозування трьох результатів: Story Points, тривалості виконання задачі та рівня ризику. Для прогнозування Story Points і тривалості виконання застосовано регресійні моделі, а для визначення рівня ризику — класифікаційну модель. У роботі

використано алгоритми Random Forest Regressor та Random Forest Classifier, оскільки вони добре працюють з табличними даними, підтримують роботу з числовими й категоріальними ознаками та забезпечують достатню точність без надмірної складності налаштування [18, 21, 22].

Data Analytics Module відповідає за візуальний аналіз даних. Він дозволяє переглядати розподіл задач за рівнем ризику, Story Points, пріоритетами, командами, типами задач, а також аналізувати взаємозв'язки між числовими ознаками. У системі реалізовано стовпчикові діаграми, scatter plot, boxplot і матрицю кореляцій. Це дає змогу не лише отримати прогноз для конкретної задачі, а й оцінити загальну структуру навчального датасету.

Загальна логіка роботи системи полягає в тому, що користувач вводить короткий опис задачі та задає її параметри в інтерфейсі. Далі Semantic Engine аналізує текст і формує структуровану задачу, а Predictive Engine перетворює параметри у формат, придатний для ML-моделей, і прогнозує Story Points, тривалість виконання та рівень ризику. Після цього система формує рекомендацію для Project або Product Manager, а модуль аналітики відображає графіки й узагальнену статистику за датасетом.

У цій архітектурі (рис. 2.1) користувач взаємодіє із системою через Streamlit-інтерфейс. Дані, які вводяться користувачем, передаються до модулів обробки. Semantic Engine відповідає за текстову частину, Predictive Engine — за прогнозування, а Data Analytics Module працює з датасетом і дозволяє переглядати узагальнені закономірності. Функціонально система приймає короткий опис задачі, визначає її тип і компонент, формує опис, User Story та Acceptance Criteria, прогнозує Story Points, тривалість і ризик, а також надає рекомендацію менеджеру.

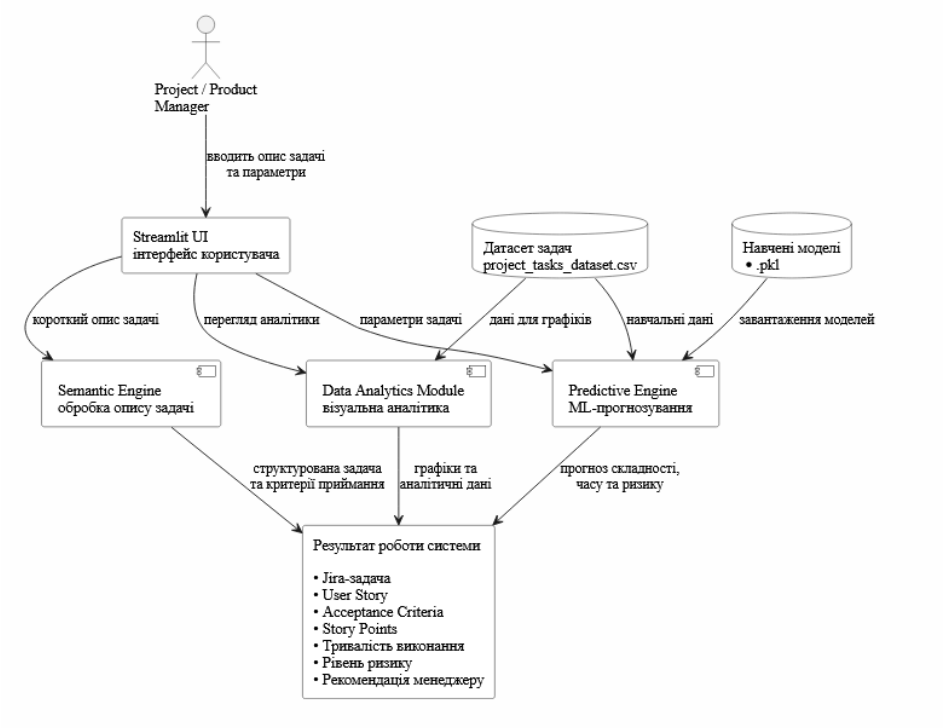

Рисунок 2.1 – Архітектура програмного прототипу AI-асистента

З погляду користувача система працює як інтерактивний помічник. Product Manager або Project Manager може ввести коротку ідею задачі, після чого система автоматично сформує структурований опис, критерії приймання, можливі ризики та прогнозовані параметри. Таким чином, запропонована архітектура дозволяє реалізувати гібридний підхід до підтримки діяльності Project та Product Manager, поєднуючи текстову обробку, прогнозування та аналітику даних в одному програмному прототипі.

2.2. Формування датасету, вибір ознак і підготовка даних для моделей машинного навчання

Для навчання моделей машинного навчання необхідні історичні дані про задачі програмних проєктів. У реальних умовах такі дані можуть бути отримані з Jira, GitHub Issues, Linear, ClickUp або інших систем управління задачами. Вони можуть містити опис задачі, тип задачі, пріоритет,

відповідальну команду, оцінку складності, фактичну тривалість виконання, кількість залежностей, статуси, коментарі та інші параметри. Однак у межах дипломної роботи доступ до реальних корпоративних даних був обмежений, тому було прийнято рішення сформувати синтетичний датасет, який імітує типову структуру історичних даних програмного проєкту.

Використання синтетичного датасету є прийнятним для прототипування, оскільки дозволяє перевірити логіку роботи моделей, структуру системи та можливість прогнозування на основі формалізованих ознак. При цьому дані не є випадковим набором значень, а побудовані з урахуванням логіки, близької до реального процесу оцінювання задач в Agile-командах. Під час генерації враховувалися фактори, які можуть впливати на складність, тривалість виконання та рівень ризику задачі.

Датасет було сформовано у файлі dataset_generator.py. У результаті виконання цього файлу створюється CSV-файл data/project_tasks_dataset.csv. Кількість записів у датасеті становить 1000 задач. Кожен запис відповідає окремій задачі програмного проєкту та містить як вхідні ознаки, що використовуються для навчання моделей, так і цільові змінні, які необхідно прогнозувати.

Структуру датасету наведено в таблиці 2.1.

Таблиця 2.1 – Основні поля датасету задач програмного проєкту

Поле	**Тип даних**	**Опис**
task_id	числовий	Унікальний номер задачі
task_description	текстовий	Короткий опис задачі
team	категоріальний	Команда, відповідальна за виконання
priority	категоріальний	Пріоритет задачі: Low, Medium, High, Critical

Продовження таблиці 2.1.

task_type	категоріальний	Тип задачі: Feature, Bug, Task, Improvement, Research
component	категоріальний	Компонент системи
complexity	числовий	Складність задачі за шкалою від 1 до 10
dependencies	числовий	Кількість залежностей
team_size	числовий	Розмір команди
sprint_day	числовий	День спринту
backlog_age_days	числовий	Вік задачі у беклозі
developer_experience	числовий	Рівень досвіду розробника
has_external_integration	бінарний	Наявність зовнішньої інтеграції
requires_design	бінарний	Потреба в дизайні
requires_testing	бінарний	Потреба в тестуванні
team_capacity_hours	числовий	Орієнтовна місткість команди
story_points	числовий	Цільова змінна для регресії
duration_hours	числовий	Цільова змінна для регресії
delay_probability	числовий	Ймовірність затримки
risk_level	категоріальний	Цільова змінна для класифікації

Під час генерації даних було враховано логіку, за якою складніші задачі мають більшу кількість Story Points. На цей показник впливають складність задачі, кількість залежностей, пріоритет, тип задачі, потреба в зовнішній

інтеграції, необхідність тестування та досвід розробника. Наприклад, задача з критичним пріоритетом, великою кількістю залежностей і зовнішньою інтеграцією отримує вищу оцінку складності. Водночас високий рівень досвіду розробника може зменшувати очікувану складність або тривалість виконання.

Орієнтовна тривалість виконання задачі також формується на основі декількох факторів. Базовим параметром є Story Points, після чого враховуються залежності, інтеграції, потреба в тестуванні, розмір команди, досвід розробника та день спринту. Ймовірність затримки розраховується з урахуванням ризик-факторів: високої складності, великої тривалості, значної кількості залежностей, критичного або високого пріоритету, зовнішньої інтеграції, потреби в тестуванні та недостатнього досвіду розробника. Після цього задача відноситься до одного з трьох класів ризику: Low, Medium або High.

Для забезпечення відтворюваності результатів у файлі генерації датасету було зафіксовано значення випадкового стану RANDOM_SEED = 42. Це дозволяє при повторному запуску отримувати однаковий датасет і, відповідно, однакові результати навчання моделей. Такий підхід є важливим для експериментальної частини дипломної роботи, оскільки забезпечує стабільність результатів і можливість їх повторної перевірки.

Після генерації датасету виконується підготовка даних для машинного навчання. У роботі було виділено набір вхідних ознак FEATURE_COLUMNS, які використовуються для навчання моделей. Текстове поле task_description у поточній реалізації не використовується безпосередньо для ML-прогнозування, оскільки основний акцент зроблено на табличних ознаках задачі. Водночас текстовий опис використовується Semantic Engine для визначення типу задачі, компонента системи та формування структурованого опису.

Ознаки було поділено на категоріальні та числові. До категоріальних належать команда, пріоритет, тип задачі та компонент системи. До числових належать складність, кількість залежностей, розмір команди, день спринту, вік задачі у беклозі, досвід розробника, наявність зовнішньої інтеграції, потреба в дизайні, потреба в тестуванні та місткість команди. Для категоріальних ознак використовується OneHotEncoder, який перетворює текстові категорії на числові бінарні ознаки. Для числових ознак застосовується StandardScaler, що дозволяє привести значення до єдиного масштабу та зменшити вплив різних діапазонів даних.

Підготовка даних реалізована за допомогою ColumnTransformer, який дозволяє одночасно застосовувати різні способи обробки до різних типів ознак. Далі ця обробка об’єднується з моделлю в єдиний Pipeline. Такий підхід є зручним, оскільки дає змогу зберігати не лише модель, а й усі етапи попередньої обробки даних в одному файлі. Отже, сформований датасет і підготовлені ознаки забезпечують основу для навчання моделей прогнозування Story Points, тривалості виконання задачі та рівня ризику.

2.3. Вибір програмного забезпечення, бібліотек і реалізація основних модулів системи

Для практичної реалізації програмного прототипу було обрано мову програмування Python. Такий вибір обґрунтований тим, що Python має розвинену екосистему бібліотек для аналізу даних, машинного навчання, візуалізації та швидкої розробки інтерактивних вебзастосунків. Python широко використовується в задачах штучного інтелекту, обробки даних і прототипування інтелектуальних систем [18, 21, 22].

Розробка проєкту виконувалась у середовищі Visual Studio Code. Для запуску системи було створено окрему структуру проєкту (рис. 2.2), яка

включає файли `dataset_generator.py`, `train_model.py`, `app.py`, `requirements.txt`, а також папки `data` і `models`.

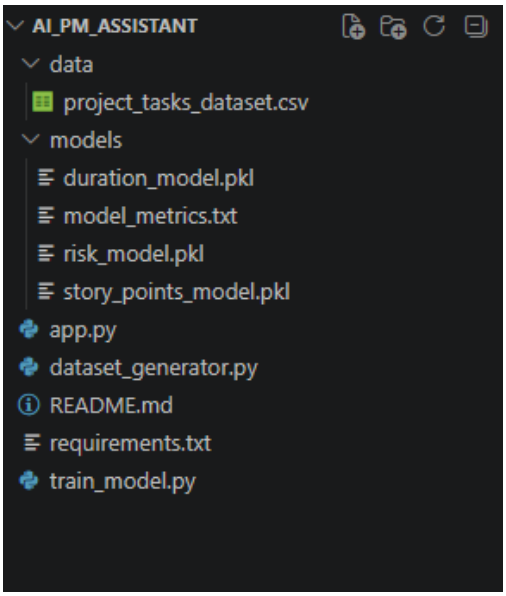

Рисунок 2.2 – Структура проєкту

Файл `dataset_generator.py` відповідає за створення синтетичного датасету задач програмного проєкту. У ньому описано логіку генерації вхідних параметрів задачі, розрахунку Story Points, тривалості виконання, ймовірності затримки та рівня ризику. Після запуску цього файлу створюється CSV-файл `project_tasks_dataset.csv` у папці `data`.

Файл `train_model.py` використовується для навчання моделей машинного навчання. Він завантажує датасет, виконує попередню обробку даних, навчає моделі прогнозування Story Points, тривалості виконання та рівня ризику, обчислює метрики якості та зберігає навчені моделі у папку `models`. Файл `app.py` містить реалізацію користувацького інтерфейсу та основну логіку роботи системи: завантаження моделей, семантичний аналіз тексту, генерацію задачі, прогнозування параметрів, формування

рекомендацій і відображення графіків. Файл `requirements.txt` містить перелік бібліотек, необхідних для запуску проєкту.

Для роботи з табличними даними використано бібліотеку Pandas, яка застосовується для створення, збереження, завантаження та обробки датасету. Бібліотека NumPy використовується для числових обчислень. Для реалізації моделей машинного навчання використано Scikit-learn, що забезпечує інструменти для поділу даних на навчальну та тестову вибірки, попередньої обробки ознак, побудови моделей, створення pipeline та оцінювання якості прогнозування.

Для прогнозування Story Points і тривалості виконання задачі використано модель RandomForestRegressor. Вона навчається на вхідних ознаках задачі та прогнозує числове значення. Для класифікації рівня ризику використано модель RandomForestClassifier, яка визначає один із трьох класів: Low, Medium або High. Для часткового врахування дисбалансу класів у моделі класифікації використано параметр `class_weight="balanced"`.

Для збереження навчених моделей застосовано бібліотеку Joblib. Вона дозволяє зберігати моделі у форматі .pkl і надалі завантажувати їх у Streamlit-застосунок без повторного навчання. Це робить роботу системи швидшою та зручнішою, оскільки користувацький інтерфейс використовує вже підготовлені моделі.

Для реалізації вебінтерфейсу було використано Streamlit. Цей інструмент дозволяє швидко створити інтерактивний вебзастосунок для роботи з моделями машинного навчання без необхідності окремо розробляти frontend і backend. У межах системи Streamlit використовується для створення лівої панелі параметрів, вкладок, кнопок, метрик, таблиць, текстових блоків і графіків.

Користувацький інтерфейс складається з чотирьох основних вкладок: AI-асистент задач, Метрики моделей, Аналітика даних і Про прототип. На

першій вкладці користувач вводить короткий опис задачі та задає параметри, після чого система формує Jira-задачу, User Story, Acceptance Criteria, ризики, рекомендацію, прогнозовані Story Points, тривалість виконання, рівень ризику та ймовірність затримки. Вкладка з метриками відображає якість навчених моделей, вкладка аналітики містить графіки за всім датасетом, а вкладка про прототип коротко описує призначення системи та її модулі.

Для побудови графіків використано засоби Streamlit і бібліотеку Matplotlib. У системі реалізовано стовпчикові діаграми, scatter plot, line chart, boxplot і матрицю кореляцій. Ці візуалізації дозволяють аналізувати розподіл задач за рівнем ризику, Story Points, пріоритетом, командою та типом задачі, а також оцінювати залежність між Story Points, тривалістю виконання та іншими числовими ознаками.

Окрему увагу в системі приділено формуванню рекомендацій для Project або Product Manager. Якщо модель прогнозує високий ризик, система рекомендує додатково переглянути задачу, уточнити залежності, розділити її на менші підзадачі або зарезервувати додатковий ресурс команди. Якщо ризик середній, система рекомендує контролювати виконання задачі та уточнити критерії приймання. Якщо ризик низький, задача може бути включена до спринту в межах стандартного планування.

Загальна логіка роботи програмного прототипу полягає в тому, що система завантажує навчені моделі, приймає від користувача текст задачі та її параметри, автоматично визначає тип задачі й компонент системи, формує набір ознак для моделей машинного навчання, прогнозує Story Points, тривалість і рівень ризику, після чого генерує текстову частину Jira-задачі, ризики та рекомендацію. Результат відображається в інтерфейсі у зручному для менеджера вигляді.

З практичного погляду система демонструє можливість автоматизації процесу формування та попереднього оцінювання задач у програмному

проєкті. Вона не замінює Project Manager або Product Manager, але може бути використана як допоміжний інструмент для скорочення часу на підготовку задач, підвищення структурованості вимог і виявлення ризиків на ранньому етапі.

Висновки до розділу 2

У другому розділі розроблено та описано програмний прототип AI-асистента для Project і Product менеджерів, що базується на поєднанні семантичної обробки тексту, машинного навчання та візуальної аналітики.

Архітектура рішення включає три основні модулі: Semantic Engine (формування структурованої задачі, User Story, Acceptance Criteria та ризиків), Predictive Engine (прогнозування Story Points, тривалості та рівня ризику) і Data Analytics Module (візуальний аналіз датасету).

Для навчання моделей сформовано синтетичний датасет на 1000 задач, що імітує історичні дані проєкту за низкою параметрів (команда, пріоритет, складність, досвід розробників тощо). Цільовими змінними виступають Story Points, тривалість виконання, ймовірність затримки та рівень ризику.

Систему реалізовано мовою Python із використанням сучасних бібліотек (Pandas, Scikit-learn, Streamlit та ін.). Для прогнозування Story Points і тривалості використано RandomForestRegressor, а для класифікації ризику — RandomForestClassifier. Моделі інтегровано у Streamlit-застосунок для отримання прогнозів у реальному часі.

Отже, реалізований прототип підтверджує можливість і доцільність застосування AI/ML-підходів для автоматизації частини управлінських задач у програмних проєктах.

3. ТЕСТУВАННЯ, ОЦІНКА ЕФЕКТИВНОСТІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

3.1. Методика тестування програмного прототипу та сценарії використання системи

Після реалізації програмного прототипу AI-асистента для Project та Product Management було проведено тестування його функціональних можливостей, моделей машинного навчання та модуля візуальної аналітики. Основною метою тестування було перевірити, чи здатна система виконувати поставлені задачі: формувати структуровану Jira-задачу, генерувати User Story та Acceptance Criteria, прогнозувати Story Points, орієнтовну тривалість виконання, рівень ризику та ймовірність затримки.

Тестування проводилося у декілька етапів. Спочатку було перевірено коректність генерації датасету, далі — навчання моделей машинного навчання, після цього — запуск Streamlit-застосунку та перевірка роботи користувацького інтерфейсу. Окремо було проаналізовано якість моделей і побудовані графіки для візуального аналізу даних.

Загальна послідовність тестування програмного прототипу була такою:

1. запуск генератора датасету `dataset_generator.py`;
2. перевірка створення файлу `project_tasks_dataset.csv`;
3. запуск файлу `train_model.py` для навчання моделей;
4. перевірка створення файлів моделей у папці `models`;
5. запуск Streamlit-застосунку через файл `app.py`;
6. введення тестових задач;
7. аналіз отриманих прогнозів і рекомендацій;
8. перевірка вкладок із метриками моделей та аналітикою датасету.

На першому етапі було виконано: `python dataset_generator.py`

У результаті було створено датасет `project_tasks_dataset.csv`, який містить 1000 записів. Кожен запис відповідає окремій задачі програмного проєкту та містить набір ознак, необхідних для навчання моделей машинного навчання. До таких ознак належать команда, пріоритет, тип задачі, компонент системи, складність, кількість залежностей, розмір команди, день спринту, вік задачі у беклозі, досвід розробника, потреба в зовнішній інтеграції, дизайні та тестуванні.

Після створення датасету було виконано навчання моделей за допомогою команди: `python train_model.py`

У результаті виконання цього файлу було навчено три моделі:

- модель прогнозування Story Points;
- модель прогнозування тривалості виконання задачі;
- модель класифікації рівня ризику.

Навчені моделі було збережено у папці models у форматі .pkl:

```
story_points_model.pkl
duration_model.pkl
risk_model.pkl
```

Також було сформовано файл `model_metrics.txt`, у якому збережено метрики якості моделей. Цей файл використовується у вкладці “Метрики моделей” Streamlit-застосунку.

Після навчання моделей було виконано запуск користувацького інтерфейсу командою: `streamlit run app.py`

Після запуску відкривається вебінтерфейс системи, у якому користувач може ввести короткий опис задачі та налаштувати параметри. Інтерфейс містить ліву панель із параметрами задачі та основну область із вкладками:

- AI-асистент задач;
- Метрики моделей;
- Аналітика даних;
- Про прототип.

На вкладці AI-асистент задач користувач вводить короткий опис задачі, після чого система автоматично визначає тип задачі, компонент системи та потребу в зовнішній інтеграції. Далі користувач може змінити значення параметрів вручну. Після натискання кнопки “Згенерувати та спрогнозувати” система формує повний результат.

Для перевірки роботи системи було використано декілька сценаріїв. Перший сценарій стосувався задачі з реалізації авторизації через Google OAuth для мобільного застосунку. Вхідні параметри цього сценарію наведено в таблиці 3.1.

Таблиця 3.1 – Вхідні параметри тестового сценарію 1

Параметр	**Значення**
Опис задачі	Реалізувати авторизацію через Google Oauth для мобільного застосунку
Відповідальна команда	Backend
Пріоритет	Високий
Тип задачі	Нова функція
Компонент системи	Автентифікація
Складність задачі	6
Кількість залежностей	2

Продовження таблиці 3.1

Розмір команди	5
День спринту	4
Вік задачі у беклозі	30 днів
Рівень досвіду розробника	6
Потреба в зовнішній інтеграції	Так
Потреба в UI/UX-дизайні	Ні
Потреба в тестуванні	Так

За цим сценарієм система сформувала структуровану Jira-задачу, User Story, Acceptance Criteria, перелік ризиків і рекомендацію для менеджера. Також було отримано такі прогнозовані результати:

- Story Points — 13;
- орієнтовна тривалість — 83,9 год;
- рівень ризику — середній;
- ймовірність затримки — 65%.

Цей результат свідчить про те, що система визначила задачу як складну, але не критично ризикову. Такий прогноз є логічним, оскільки задача пов’язана з автентифікацією та зовнішньою інтеграцією Google OAuth, однак команда має середній рівень ресурсів і достатній рівень досвіду розробника.

Другий сценарій було сформовано для перевірки роботи системи на більш ризиковій задачі. У ньому було використано параметри критичної помилки в модулі платежів. Вхідні параметри для цього сценарію наведено в таблиці 3.2.

Таблиця 3.2 – Вхідні параметри тестового сценарію 2

Параметр	**Значення**
Опис задачі	Виправити помилку списання коштів під час повторної оплати замовлення
Відповідальна команда	Backend
Пріоритет	Критичний
Тип задачі	Помилка
Компонент системи	Платежі
Складність задачі	9
Кількість залежностей	4
Розмір команди	3
День спринту	10
Вік задачі у беклозі	15 днів
Рівень досвіду розробника	4
Потреба в зовн. інтеграції	Так
Потреба в UI/UX-дизайні	Ні
Потреба в тестуванні	Так

Для цього сценарію система отримала такі результати:

- Story Points — 13;
- орієнтовна тривалість — 107,3 год;
- рівень ризику — високий;

- ймовірність затримки — 95%.

Отриманий результат є обґрунтованим, оскільки задача має критичний пріоритет, високу складність, багато залежностей, малий розмір команди, потребує зовнішньої інтеграції та стосується модуля платежів. Такі задачі у реальних IT-проєктах зазвичай потребують підвищеної уваги з боку Project Manager, оскільки можуть впливати на фінансові операції, безпеку та довіру користувачів.

Третій сценарій може бути використаний для перевірки задачі з нижчим рівнем ризику. Наприклад, покращення адаптивності карток товарів у мобільній версії каталогу. Така задача має нижчу складність, меншу кількість залежностей і не потребує зовнішньої інтеграції. У цьому випадку очікується нижчий або середній рівень ризику, залежно від обраних параметрів.

Тестування показало, що зміна параметрів у лівій панелі впливає на результати у вкладці AI-асистент задач. При збільшенні складності, кількості залежностей, пріоритету або потреби в зовнішній інтеграції система прогнозує більшу тривалість, вищий рівень ризику та більшу ймовірність затримки. Це свідчить про логічну узгодженість роботи прототипу.

Водночас вкладки Метрики моделей та Аналітика даних не змінюються при зміні параметрів поточної задачі. Це є коректною поведінкою системи, оскільки ці вкладки відображають не результат конкретного прогнозу, а загальну інформацію про якість моделей і структуру всього датасету. Метрики моделей розраховуються під час навчання, а аналітичні графіки будуються на основі 1000 записів датасету.

Таким чином, методика тестування дозволила перевірити як окремі технічні етапи роботи системи, так і кінцевий сценарій використання прототипу Project або Product Manager.

3.2. Оцінка якості моделей прогнозування Story Points, тривалості виконання та рівня ризику

Оцінка якості моделей машинного навчання є важливим етапом перевірки працездатності розробленого програмного прототипу. У межах роботи було навчено три моделі: дві регресійні моделі та одну класифікаційну. Регресійні моделі використовуються для прогнозування числових значень, а саме Story Points і тривалості виконання задачі. Класифікаційна модель використовується для визначення рівня ризику задачі.

Для оцінювання регресійних моделей було використано такі метрики:

- MAE — середня абсолютна помилка;
- MSE — середньоквадратична помилка;
- RMSE — корінь із середньоквадратичної помилки;
- R2 Score — коефіцієнт детермінації.

Метрика MAE показує середній абсолютний розмір помилки прогнозування. Вона є зручною для інтерпретації, оскільки має ту саму одиницю виміру, що й цільова змінна. Наприклад, якщо MAE для Story Points дорівнює 0,7192, це означає, що в середньому прогноз моделі відхиляється менш ніж на 1 Story Point.

Метрика MSE сильніше штрафує великі помилки, оскільки підносить їх до квадрату. RMSE є коренем із MSE і також має одиницю виміру цільової змінної. R2 Score показує, яку частку варіативності цільової змінної пояснює модель. Значення R2 ближче до 1 свідчить про вищу якість моделі.

Для класифікаційної моделі було використано такі метрики:

- Accuracy;
- precision;
- recall;

- f1-score;
- confusion matrix.

Метрика Accuracy показує частку правильно класифікованих задач серед усіх задач тестової вибірки. Precision показує, наскільки точними є прогнози певного класу. Recall показує, яку частину реальних об’єктів цього класу модель змогла правильно знайти. F1-score є узагальненою метрикою, яка враховує і precision, і recall.

Після навчання моделей було отримано такі результати (табл. 3.3).

Таблиця 3.3 – Метрики якості регресійних моделей

Модель	**MAE**	**MSE**	**RMSE**	**R2**
Story Points Regression Model	0,7192	1,0642	1,0316	0,8496
Duration Hours Regression Model	12,5774	241,5868	15,5431	0,6891

Результати моделі прогнозування Story Points можна оцінити як хороші для дипломного прототипу. Значення R2 = 0,8496 означає, що модель пояснює приблизно 85% варіативності цільової змінної. Середня абсолютна помилка MAE = 0,7192 свідчить, що модель у середньому помиляється менш ніж на 1 Story Point. Для задач попереднього оцінювання складності це є достатньо якісним результатом.

Модель прогнозування тривалості виконання задачі показала помірно якісний результат. Значення R2 = 0,6891 означає, що модель пояснює приблизно 69% варіативності тривалості задач. Середня абсолютна помилка становить 12,5774 години. Така помилка є прийнятною для попереднього оцінювання, оскільки тривалість виконання задач у реальних програмних проєктах залежить від великої кількості факторів, зокрема людського фактора,

уточнення вимог, блокерів, комунікації між командами, змін пріоритетів і технічних складностей.

Порівняно з прогнозуванням Story Points, прогнозування тривалості є складнішою задачею. Story Points відображають відносну складність, тоді як тривалість у годинах залежить від конкретної команди, досвіду виконавця, робочого навантаження, контексту спринту та непередбачуваних обставин. Тому нижче значення R2 для моделі тривалості є очікуваним.

Оцінка класифікаційної моделі ризику наведена в таблиці 3.4.

Таблиця 3.4 – Метрики якості моделі класифікації рівня ризику

Клас ризику	**Precision**	**Recall**	**F1-score**	**Support**
High	1,00	0,35	0,52	34
Low	1,00	0,28	0,44	32
Medium	0,75	1,00	0,86	134
Accuracy			0,7750	200

Загальна точність класифікаційної моделі становить 0,7750, тобто модель правильно класифікувала 77,5% задач тестової вибірки. Для дипломного прототипу це є прийнятним результатом. Водночас classification report показує, що модель має певну особливість: вона дуже добре визначає клас Medium, але обережніше розпізнає класи High і Low.

Precision для класів High і Low дорівнює 1,00. Це означає, що коли модель прогнозує ці класи, вона не помиляється. Однак recall для цих класів є нижчим: 0,35 для High і 0,28 для Low. Це означає, що модель знаходить лише

частину задач, які насправді належать до високого або низького ризику. Значну частину таких задач модель відносить до класу Medium.

Ця особливість пояснюється дисбалансом класів у датасеті. У навчальних даних найбільше задач мають середній рівень ризику, тому модель частіше прогнозує саме цей клас. Це є типовою ситуацією для задач класифікації, коли один із класів представлений значно більшою кількістю прикладів.

Матриця помилок класифікаційної моделі має такий вигляд:

```
[[ 12   0  22]
 [  0   9  23]
 [  0   0 134]]
```

У цій матриці видно, що:

- з 34 задач класу High модель правильно визначила 12, а 22 віднесла до Medium;
- з 32 задач класу Low модель правильно визначила 9, а 23 віднесла до Medium;
- усі 134 задачі класу Medium були визначені правильно.

Отже, модель ризику працює коректно, однак має тенденцію до переважного прогнозування середнього рівня ризику. У реальних умовах для виробничого використання таку модель доцільно було б додатково покращити, зокрема збалансувати класи, збільшити кількість прикладів високого та низького ризику, а також оптимізувати recall для класу High. Це важливо тому, що в реальному управлінні проєктами критично не пропустити високоризикову задачу.

Проте в межах бакалаврської дипломної роботи отриманий результат є достатнім для демонстрації працездатності підходу. Система показує, що на

основі параметрів задачі можна будувати прогноз її складності, тривалості та рівня ризику. Отримані метрики підтверджують доцільність використання моделей машинного навчання як допоміжного інструмента для Project та Product Manager.

3.3. Візуальний аналіз датасету та інтерпретація результатів роботи AI-асистента

Окрім прогнозування параметрів конкретної задачі, у програмному прототипі було реалізовано модуль візуальної аналітики даних. Його метою є надання користувачу загального уявлення про структуру датасету, розподіл задач, взаємозв’язки між ознаками та характер цільових змінних. Такий модуль є корисним для Project Manager і Product Manager, оскільки дозволяє оцінити загальні закономірності в задачах програмного проєкту.

У вкладці Аналітика даних відображаються основні статистичні показники датасету:

- кількість задач — 1000;
- середні Story Points — 10,79;
- середня тривалість задач — 61,74 год;
- середня ймовірність затримки — 49,4%.

Ці показники дають загальне уявлення про характер згенерованого датасету. Значення середніх Story Points свідчить про те, що в датасеті переважають задачі середньої та високої складності. Середня тривалість 61,74 год вказує на те, що значна частина задач потребує суттєвого часу на виконання. Середня ймовірність затримки 49,4% показує, що датасет містить достатню кількість задач із потенційним ризиком.

Першим графіком є розподіл задач за рівнем ризику (рис. 3.1). Він показує, що найбільшу кількість задач становить клас Medium. Класи Low і High представлені меншою кількістю записів. Саме ця особливість пояснює результати класифікаційної моделі, яка краще розпізнає середній рівень ризику. Такий розподіл є логічним, оскільки в реальних проєктах більшість задач не є повністю безризиковими або критичними, а належать до середнього рівня ризику.

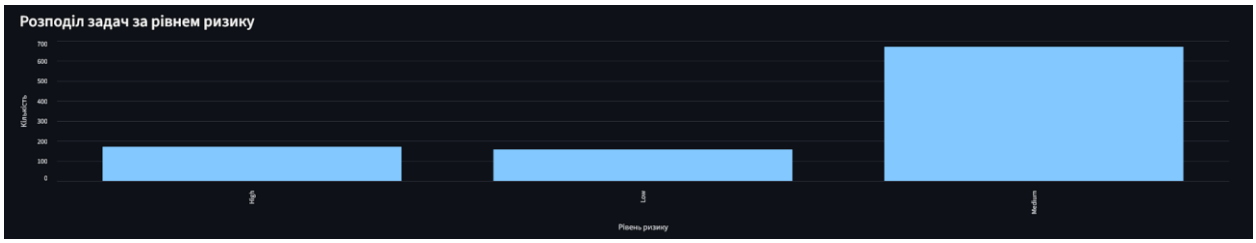

Рисунок 3.1 – Розподіл задач за рівнем ризику

Другий графік відображає розподіл задач за Story Points (рис. 3.2). На ньому видно, що значна частина задач має високу оцінку Story Points. Найбільша кількість задач зосереджена біля значення 13. Це пояснюється логікою генерації датасету, де задачі з високою складністю, залежностями, зовнішніми інтеграціями та тестуванням отримують більшу оцінку. У реальному проєкті така ситуація може свідчити про наявність великої кількості складних задач у беклозі, які потребують декомпозиції.

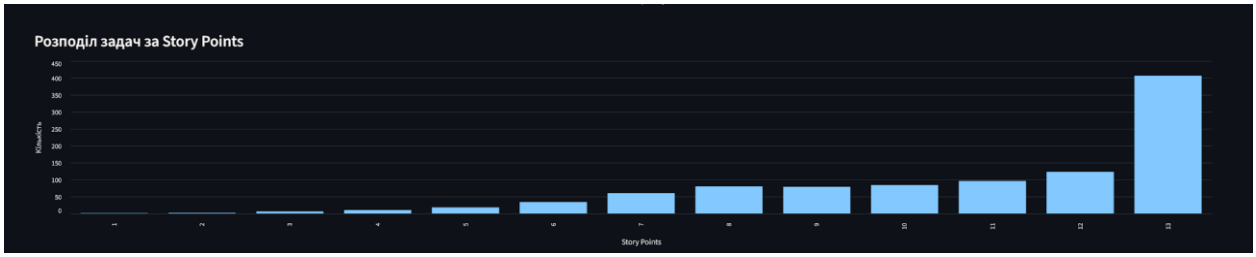

Рисунок 3.2 – Розподіл задач за Story Points

Графік середньої тривалості задач за рівнем ризику (рис. 3.3) показує, що задачі з високим рівнем ризику мають найбільшу середню тривалість виконання. Задачі з низьким ризиком мають найменшу середню тривалість, а задачі із середнім ризиком займають проміжне положення. Це підтверджує логічний зв’язок між тривалістю виконання та ризиком: чим більше часу потребує задача, тим вища ймовірність появи затримок або блокерів.

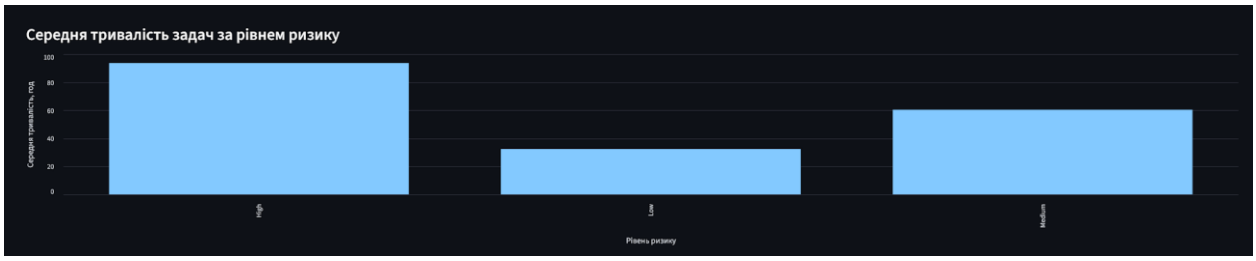

Рисунок 3.3 – Середня тривалість задач за рівнем ризику

Графік середньої тривалості задач за командами (рис. 3.4) дозволяє порівняти навантаження між різними командами. У датасеті представлені команди Backend, Frontend, QA, Design, DevOps, Fullstack, Data та Mobile. Значення середньої тривалості для команд є близькими, однак певні відмінності все ж спостерігаються. Такий графік може бути корисним для менеджера, оскільки дозволяє оцінити, які команди частіше отримують більш трудомісткі задачі.

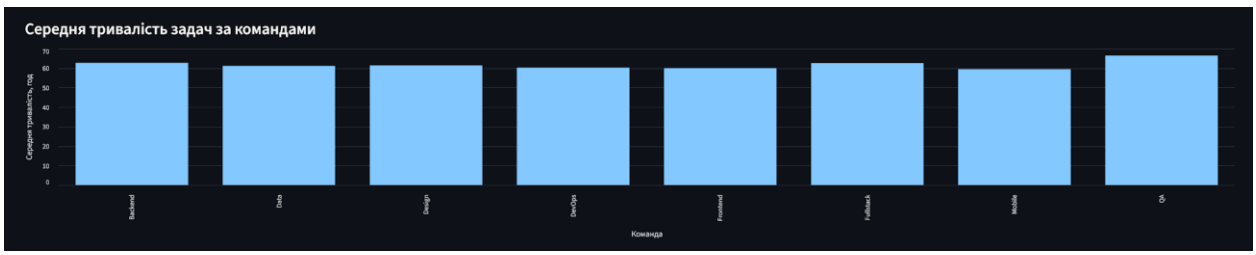

Рисунок 3.4 – Середня тривалість задач за командами

Графік кількості задач за пріоритетом (рис. 3.5) показує розподіл між пріоритетами Low, Medium, High і Critical. Наявність задач різних пріоритетів дозволяє моделі враховувати вплив бізнес-важливості задачі на її складність і ризик. У реальних умовах задачі з високим або критичним пріоритетом потребують більшого контролю з боку Project Manager, оскільки вони можуть впливати на ключові функції продукту або строки релізу.

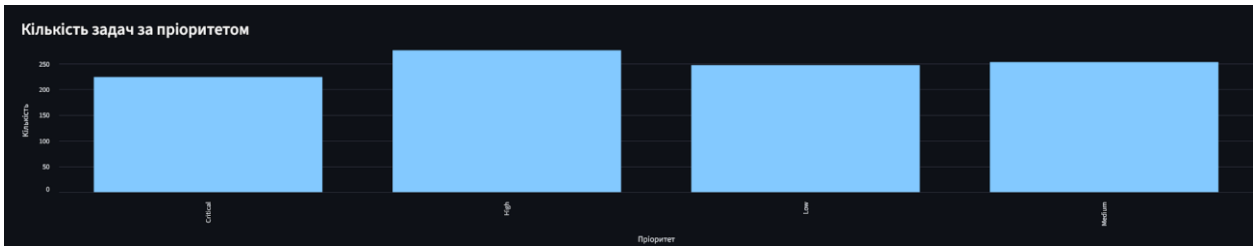

Рисунок 3.5 – Кількість задач за пріоритетом

Графік середніх Story Points за типом задачі (рис. 3.6) дозволяє порівняти складність різних типів задач: Feature, Bug, Task, Improvement і Research. У датасеті задачі типу Feature та Research зазвичай мають вищі Story Points, оскільки нова функціональність або дослідження часто потребують більше часу на уточнення, розробку та перевірку. Bug і Improvement можуть мати різну складність залежно від контексту.

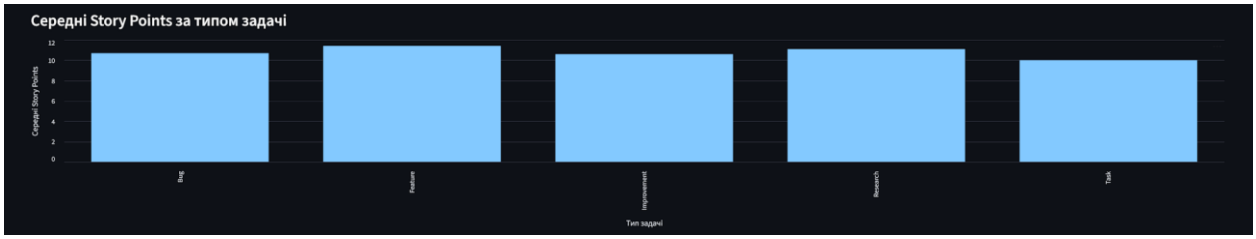

Рисунок 3.6 – Середні Story Points за типом задачі

Scatter plot залежності між Story Points і тривалістю виконання (рис. 3.7) показує загальну тенденцію: зі збільшенням Story Points збільшується і тривалість задачі. Це є очікуваним результатом, оскільки Story Points відображають складність, обсяг і невизначеність задачі. Водночас на графіку видно, що для однакових значень Story Points тривалість може відрізнятися. Це пояснюється впливом інших факторів: кількості залежностей, досвіду розробника, зовнішньої інтеграції, розміру команди та потреби в тестуванні.

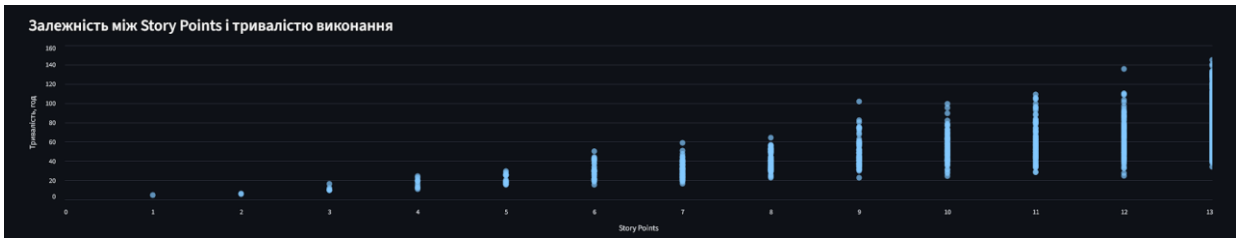

Рисунок 3.7 – Залежність між Story Points і тривалістю виконання

Line chart середньої тривалості задач залежно від дня спринту (рис. 3.8) дозволяє оцінити, як змінюється очікувана тривалість задач у межах спринту. У датасеті спостерігаються певні коливання середньої тривалості залежно від дня спринту. У реальному управлінні проєктами такі графіки можуть використовуватися для аналізу того, чи не плануються складні задачі занадто пізно в межах спринту.

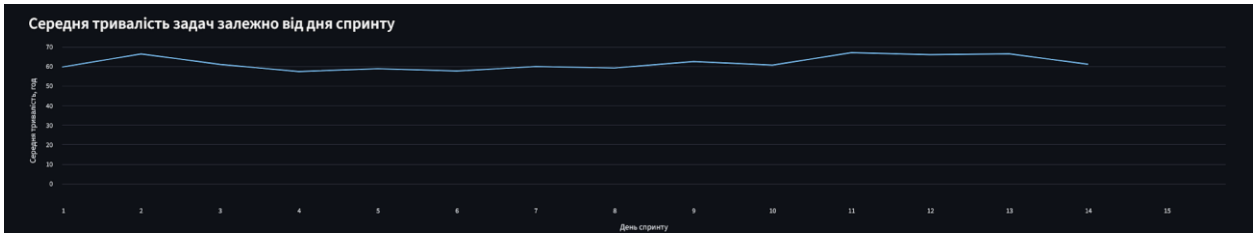

Рисунок 3.8 – Середня тривалість задач залежно від дня спринту

Boxplot розподілу тривалості задач за рівнем ризику (рис. 3.9) дозволяє оцінити не лише середні значення, а й розкид тривалості. На графіку видно, що задачі з високим ризиком мають вищу медіану тривалості та більший діапазон значень. Задачі з низьким ризиком мають меншу тривалість і менший розкид. Це підтверджує, що ризикові задачі не лише довші, а й менш передбачувані.

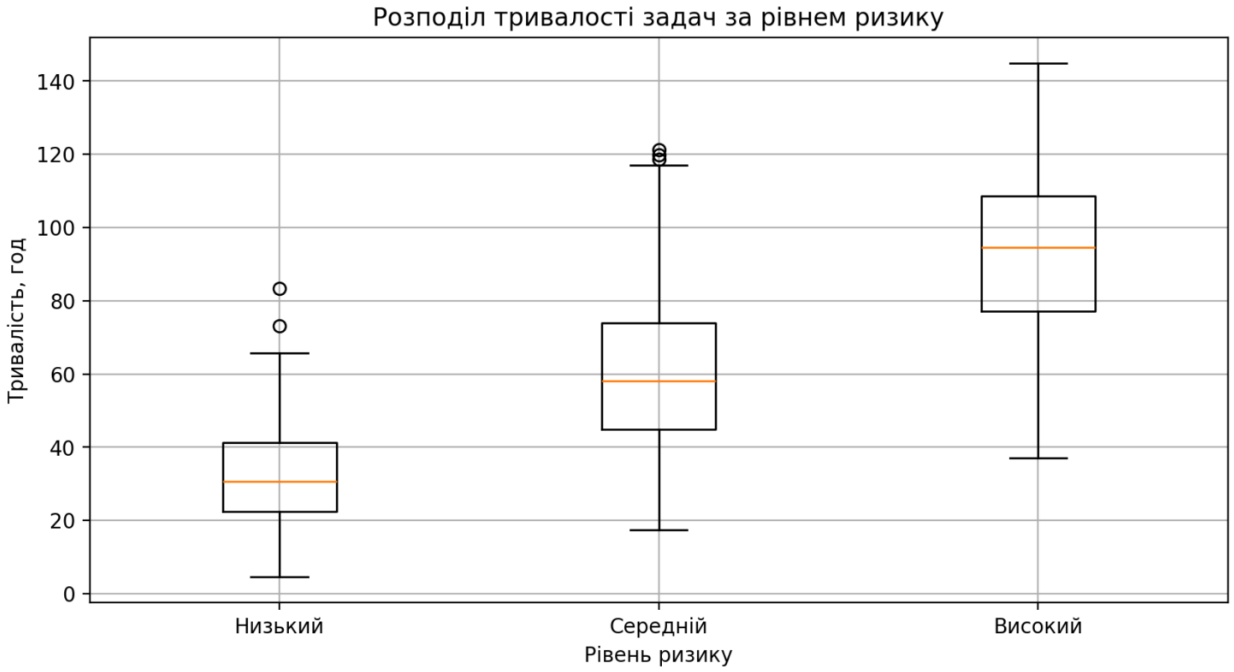

Рисунок 3.9 – Розподіл тривалості задач за рівнем ризику

Матриця кореляцій числових ознак (рис. 3.10) дозволяє проаналізувати взаємозв’язки між параметрами задач. На ній видно, що Story Points мають помітний зв’язок зі складністю, залежностями, зовнішньою інтеграцією та тривалістю. Тривалість виконання також пов’язана з ймовірністю затримки. Досвід розробника має від’ємний зв’язок із тривалістю та ймовірністю затримки, що є логічним: чим вищий досвід розробника, тим нижчий ризик затримки і менша очікувана тривалість.

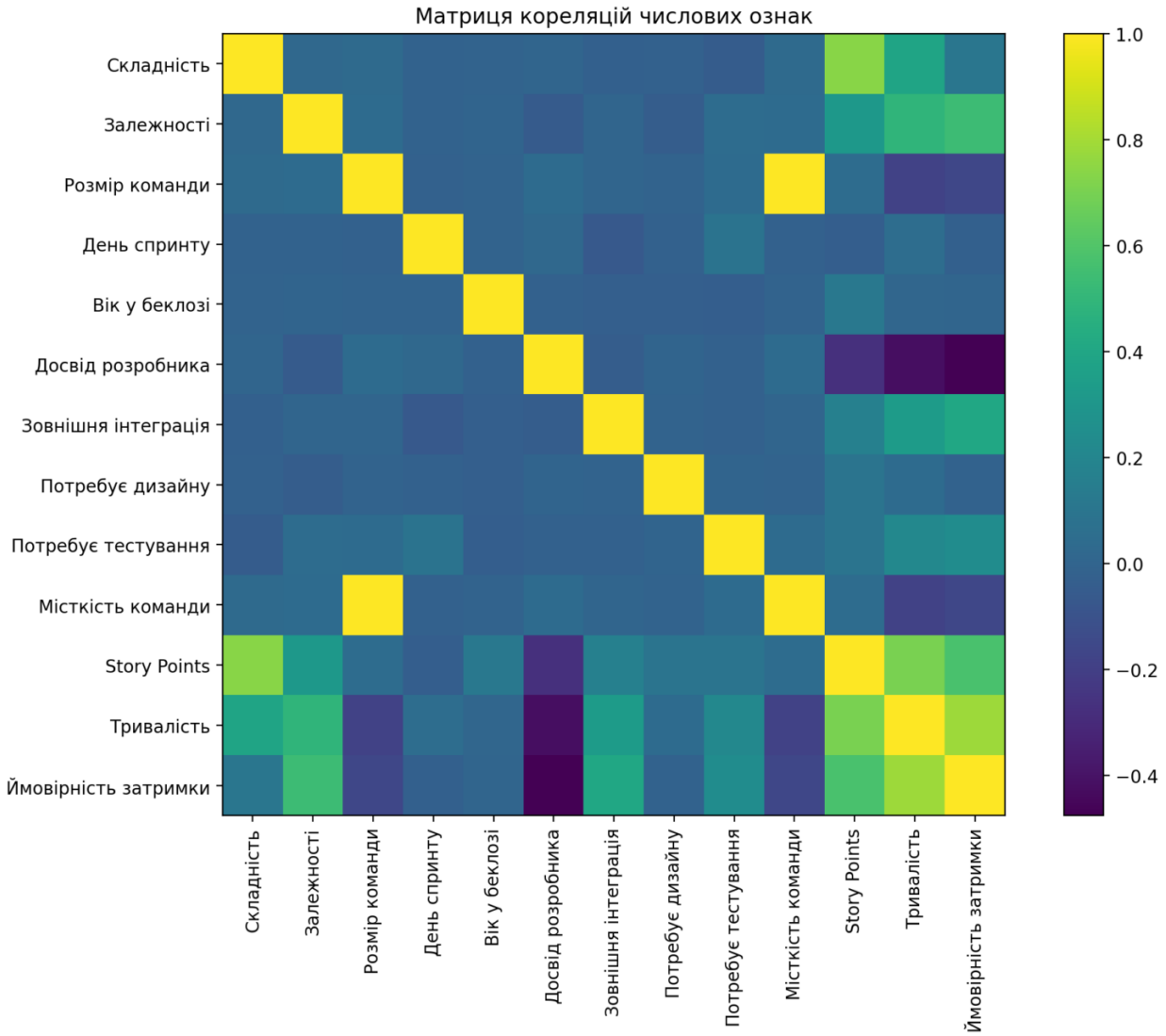

Рисунок 3.10 – Матриця кореляцій числових ознак

Окремо було проаналізовано роботу AI-асистента на тестових прикладах. У першому прикладі задача з Google OAuth отримала середній рівень ризику та ймовірність затримки 65%. Система сформувала рекомендацію про необхідність моніторингу задачі, контролю залежностей і уточнення Acceptance Criteria. Це відповідає логіці задачі, оскільки вона має зовнішню інтеграцію та стосується автентифікації.

У другому прикладі задача з виправлення критичної помилки платежів отримала високий рівень ризику та ймовірність затримки 95%. Система рекомендувала додатково переглянути задачу перед включенням у спринт, уточнити залежності, зарезервувати додаткову місткість команди та за потреби розділити задачу на менші підзадачі. Така рекомендація є обґрунтованою, оскільки задачі, пов’язані з платежами та критичними помилками, мають високий вплив на бізнес і користувачів.

Отже, візуальний аналіз підтверджує логічність сформованого датасету та результатів прогнозування. Графіки демонструють, що складність, тривалість і ризик задач пов’язані між собою. Результати роботи AI-асистента показують, що система може бути використана як допоміжний інструмент для попереднього оцінювання задач, підготовки документації та підтримки планування спринтів.

Висновки до розділу 3

У третьому розділі було проведено тестування програмного прототипу AI-асистента для Project та Product Management, оцінено якість моделей машинного навчання та проаналізовано результати роботи системи. Тестування показало, що система коректно виконує основні функції: приймає опис задачі, визначає її тип і компонент, формує структуровану Jira-задачу, генерує User Story та Acceptance Criteria, прогнозує Story Points, тривалість виконання, рівень ризику та ймовірність затримки.

Було навчено три моделі машинного навчання. Модель прогнозування Story Points показала хороший результат: MAE = 0,7192, RMSE = 1,0316, R2 = 0,8496. Це свідчить про здатність моделі достатньо точно оцінювати складність задач. Модель прогнозування тривалості виконання задачі показала помірно якісний результат: MAE = 12,5774, RMSE = 15,5431, R2 = 0,6891.

Такий результат є прийнятним для попереднього оцінювання, оскільки тривалість задач у програмних проєктах залежить від багатьох факторів.

Модель класифікації рівня ризику досягла точності Accuracy = 0,7750. Аналіз classification report показав, що модель добре визначає задачі із середнім рівнем ризику, але менш активно розпізнає класи низького та високого ризику. Це пояснюється дисбалансом класів у датасеті, де найбільшу частку становлять задачі класу Medium. У реальних умовах таку модель доцільно було б додатково покращити, однак у межах дипломної роботи вона демонструє працездатність підходу.

Було також виконано візуальний аналіз датасету. Побудовані графіки показали розподіл задач за рівнем ризику, Story Points, пріоритетами, командами, типами задач, а також залежність між Story Points і тривалістю виконання. Boxplot підтвердив, що задачі з високим ризиком мають більшу тривалість і ширший розкид значень. Матриця кореляцій показала наявність логічних зв’язків між складністю, залежностями, тривалістю, ймовірністю затримки та рівнем досвіду розробника.

Отримані результати підтверджують, що розроблений програмний прототип може використовуватися як допоміжний інструмент для Project та Product Manager. Він дозволяє автоматизувати частину рутинної роботи, скоротити час на формування задач, надати попередню оцінку складності та ризику, а також забезпечити менеджера додатковою аналітичною інформацією для планування спринту. У межах дипломної роботи система досягла поставленої мети та підтвердила доцільність використання гібридного AI/ML-підходу для підтримки управління програмними проєктами.

ВИСНОВКИ

У дипломній роботі було досліджено вплив AI-інструментів на роль Project та Product Manager у процесах управління програмними проєктами. У межах роботи розглянуто теоретичні основи застосування штучного інтелекту в Project та Product Management, проаналізовано сучасні AI-рішення, методи NLP, LLM, регресії та класифікації, а також розроблено програмний прототип AI-асистента для підтримки управлінських задач у життєвому циклі розробки програмного забезпечення.

У першому розділі було визначено, що Project Manager та Product Manager виконують важливі функції в SDLC та Agile-командах. Project Manager відповідає за планування, координацію, контроль строків, управління ризиками та ресурсами, а Product Manager — за формування цінності продукту, управління вимогами, пріоритизацію беклогу та взаємодію із зацікавленими сторонами. Встановлено, що значна частина їхньої роботи має рутинний характер і може бути частково автоматизована за допомогою AI-інструментів.

Було проаналізовано сучасні AI-рішення для підтримки управління програмними проєктами, зокрема інструменти, які допомагають формувати документацію, описувати задачі, аналізувати беклог, автоматизувати робочі процеси та підтримувати прийняття управлінських рішень. Водночас визначено, що більшість наявних рішень орієнтована переважно або на генерацію тексту, або на автоматизацію окремих операцій. Це обґрунтовує доцільність розробки гібридного підходу, який поєднує текстову генерацію, машинне навчання та аналітику даних.

У роботі було обґрунтовано використання гібридного підходу, який складається з трьох основних компонентів: Semantic Engine, Predictive Engine та Data Analytics Module. Semantic Engine відповідає за формування

структурованої Jira-задачі, User Story, Acceptance Criteria, ризиків і рекомендацій. Predictive Engine виконує прогнозування Story Points, орієнтовної тривалості виконання та рівня ризику задачі. Data Analytics Module забезпечує візуальний аналіз датасету та допомагає оцінити закономірності між параметрами задач.

Науково-практична новизна роботи полягає в тому, що запропоновано власну реалізацію гібридної AI-системи для підтримки Project та Product Manager, яка поєднує семантичну обробку опису задачі, ML-прогнозування параметрів задачі та візуальну аналітику даних в одному програмному прототипі. Запропонований підхід не претендує на абсолютну новизну напряму використання AI в управлінні проєктами, однак демонструє практичне поєднання кількох методів для комплексної підтримки управлінських рішень.

У другому розділі було розроблено архітектуру програмного прототипу AI-асистента. Система реалізована у вигляді вебзастосунку на базі Streamlit. Для практичної частини було сформовано синтетичний датасет на 1000 задач, який імітує історичні дані програмного проєкту. У датасеті враховано такі ознаки, як команда, пріоритет, тип задачі, компонент системи, складність, кількість залежностей, розмір команди, день спринту, вік задачі у беклозі, досвід розробника, потреба в зовнішній інтеграції, дизайні та тестуванні.

У межах практичної реалізації було навчено три моделі машинного навчання. Для прогнозування Story Points і тривалості виконання задачі використано RandomForestRegressor, а для класифікації рівня ризику — RandomForestClassifier. Навчені моделі збережено у форматі .pkl, що дозволяє використовувати їх у Streamlit-застосунку без повторного навчання.

У третьому розділі було проведено тестування програмного прототипу та оцінено якість роботи моделей. Модель прогнозування Story Points показала хороший результат: MAE = 0,7192, RMSE = 1,0316, R2 = 0,8496. Це свідчить

про те, що модель достатньо точно оцінює складність задач на основі заданих параметрів.

Модель прогнозування тривалості виконання задачі показала помірно якісний результат: MAE = 12,5774, RMSE = 15,5431, R2 = 0,6891. Такий результат є прийнятним для попереднього оцінювання, оскільки тривалість виконання задач у реальних IT-проєктах залежить від багатьох факторів: людського фактора, уточнення вимог, зміни пріоритетів, блокерів, комунікації між командами та технічних складностей.

Модель класифікації рівня ризику досягла точності Accuracy = 0,7750. Аналіз classification report показав, що модель найкраще визначає задачі із середнім рівнем ризику, однак обережніше розпізнає класи низького та високого ризику. Це пояснюється дисбалансом класів у датасеті, де найбільшу частку становлять задачі класу Medium. У реальних умовах таку модель доцільно було б додатково покращити шляхом балансування класів, збільшення кількості прикладів високого ризику та оптимізації recall для класу High.

Також у роботі було реалізовано модуль візуальної аналітики. Побудовано графіки розподілу задач за рівнем ризику, Story Points, пріоритетом, командами та типами задач. Додатково реалізовано scatter plot для аналізу залежності між Story Points і тривалістю виконання, line chart для оцінки середньої тривалості залежно від дня спринту, boxplot для аналізу розподілу тривалості за рівнем ризику та матрицю кореляцій числових ознак. Отримані графіки підтверджують логічні взаємозв’язки між складністю задач, тривалістю виконання, кількістю залежностей і ймовірністю затримки.

На тестових сценаріях було підтверджено працездатність системи. Наприклад, для задачі реалізації Google OAuth система сформувала Jira-задачу, User Story, Acceptance Criteria, визначила ризики та спрогнозувала 13 Story Points, 83,9 години виконання, середній рівень ризику та 65%

ймовірність затримки. Для критичної задачі, пов’язаної з помилкою в оплаті, система спрогнозувала високий рівень ризику та 95% ймовірність затримки, що є логічним з огляду на критичність, складність і наявність зовнішньої інтеграції.

Отже, поставлену мету дипломної роботи було досягнуто. Було проаналізовано вплив AI-інструментів на діяльність Project та Product Manager, обґрунтовано доцільність гібридного підходу та реалізовано програмний прототип AI-асистента, який автоматизує формування задач, прогнозує складність, тривалість і рівень ризику, а також надає рекомендації для менеджера.

Практичне значення роботи полягає в тому, що розроблений прототип може бути використаний як допоміжний інструмент для попереднього оцінювання задач, підготовки управлінської документації, аналізу ризиків і підтримки планування спринтів. Система не замінює Project Manager або Product Manager, але може зменшити обсяг рутинної роботи, підвищити структурованість задач і надати додаткову аналітичну інформацію для прийняття управлінських рішень.

У подальшому розроблений прототип можна вдосконалити шляхом підключення реальних даних із Jira, GitHub або Linear, використання великих мовних моделей для глибшої генерації документації, застосування embeddings для пошуку дублікатів задач у беклозі, покращення моделі класифікації ризиків і додавання інтеграції з реальними системами управління проєктами. Таким чином, результати роботи можуть бути основою для подальшого розвитку інтелектуальних систем підтримки Project та Product Management.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Василенко В. М., Вакалюк Т. А. Штучний інтелект в управлінні проєктами: аналіз сучасних досліджень та перспектив розвитку. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. 2024. Т. 35(74), № 4. С. 60–67. DOI: 10.32782/2663-5941/2024.4/10. URL: (https://www.tech.vernadskyjournals.in.ua/journals/2024/4_2024/12.pdf) (дата звернення: 12.03.2026).

2. Бачинський О. Використання штучного інтелекту як інструменту управління проєктами. *Економіка та суспільство*. 2024. URL: (https://economyandsociety.in.ua/index.php/journal/article/view/3734) (дата звернення: 12.03.2026).

3. Крискун І., Орлова-Курилова О. Керування ризиками в управлінні ІТ-проєктами з використанням інструментів штучного інтелекту. *Вчені записки Університету «КРОК»*. 2025. № 2(78). С. 315–325. DOI: 10.31732/2663-2209-2025-78-315-325. URL: (https://snku.krok.edu.ua/index.php/vcheni-zapiski-universitetu-krok/article/view/954) (дата звернення: 12.03.2026).

4. Насад Н. В., Крискун І. М. Засади управління проєктами ІТ-компаній: методики та інструменти. *Економіка. Менеджмент. Бізнес*. 2025. № 1. С. 57–62. DOI: 10.31673/2415-8089.2025.015762. URL: (https://journals.dut.edu.ua/index/login?source=%2Findex.php%2Femb%2Farticle%2Fdownload%2F3138%2F3019%2F%3F) (дата звернення: 18.03.2026).

5. Полуектова Н., Левицький С. Впровадження ШІ-агентів для автоматизації пріоритизації завдань в системах управління проєктами. *Цифрова економіка та інформаційні технології*. 2025. № 4.

6. Гудаков Д. Застосування штучного інтелекту для управління проєктами людино-орієнтованих інформаційних технологій. 2025. URL: (https://vottp.khmnu.edu.ua/index.php/vottp/article/view/465) (дата звернення: 18.03.2026).

7. Глибовець М. М., Задохін Д. В., Дехтяр Б.-Я., Пєчкурова О. М. Оброблення природної мови за допомоги великих мовних моделей і методів машинного навчання. 2024. DOI: 10.18523/2617-3808.2024.7.102-111. URL: (https://ekmair.ukma.edu.ua/server/api/core/bitstreams/51d02403-adf4-4ff9-8d19-a56259f0346f/content) (дата звернення: 18.03.2026).

8. Переяславська С., Смагіна О. Вивчення методів обробки природної мови для створення інтелектуальних систем, таких як чат-боти та системи автоматичного перекладу. *Herald of Khmelnytskyi National University. Technical Sciences*. 2025. Т. 349, № 2. С. 100–108. DOI: 10.31891/2307-5732-2025-349-14.

9. Зайцева О. І., Парій О. В. Трансформація ролі продакт-менеджера в умовах цифровізації економіки. *Таврійський науковий вісник. Серія: Економіка*. 2025. № 25. С. 88–97. DOI: 10.32782/2708-0366/2025.25.9. URL: (https://tnv-econom.ksauniv.ks.ua/index.php/journal/article/view/701) (дата звернення: 18.03.2026).

10. Юрчук Н. П., Кіпоренко С. С. Методології управління ІТ-проєктами: критерії вибору для ефективного управління. *Ефективна економіка*. 2024. № 11. DOI: 10.32702/2307-2105.2024.11.76. URL: (https://www.nayka.com.ua/index.php/ee/article/view/5136) (дата звернення: 18.03.2026).

11. Сметанюк О. А., Бондарчук А. В. Особливості системи управління проєктами в ІТ-компаніях. *Агросвіт*. 2020. № 10. С. 105–111. URL:

(https://www.agrosvit.info/?i=14&op=1&z=3205) (дата звернення: 28.03.2026).

12. Койбічук В. В., Єфіменко А. Ю. Ефективний менеджмент ІТ-проєктів: моделі та інструментарій. *Інвестиції: практика та досвід*. 2024. № 11. С. 142–147. DOI: 10.32702/2306-6814.2024.11.142. URL: (https://www.nayka.com.ua/index.php/investplan/article/view/3881) (дата звернення: 28.03.2026).

13. Супруненко С. А. Сучасні методичні підходи до проєктного менеджменту та аналізу. *Ефективна економіка*. 2024. № 7. DOI: 10.32702/2307-2105.2024.7.65. URL: (https://www.nayka.com.ua/index.php/ee/article/view/4241) (дата звернення: 28.03.2026).

14. Крискун І. Ключові засади управління проєктами в галузі інформаційних технологій в умовах цифрової трансформації. *Вчені записки Університету «КРОК»*. 2025. URL: (https://snku.krok.edu.ua/index.php/vcheni-zapiski-universitetu-krok/article/view/1131) (дата звернення: 28.03.2026).

15. Кондратенко О. І. Управління ризиками ІТ-проєктів на засадах ризик-орієнтованого підходу: монографія. 2025. 235 с. DOI: 10.5281/zenodo.15025658. URL: (https://zenodo.org/records/15025658) (дата звернення: 08.04.2026).

16. Грабіна К. В., Шендрик В. В. Метод управління ризиками ІТ-проєктів з врахуванням загроз та можливостей. *Управління розвитком складних систем*. 2023. № 55. С. 18–28. DOI: 10.32347/2412-9933.2023.55.18-28. URL: (https://mdcs.knuba.edu.ua/article/view/291119) (дата звернення: 08.04.2026).

17. Трушкіна Н. Управління проєктними ризиками у системі забезпечення економічної безпеки підприємств IT-індустрії. 2023. URL: (https://ndc-ipr.org/media/publications/files/Трушкіна_Н._Стаття_ж._ЕтаС_вип.__55_2023.pdf) (дата звернення: 08.04.2026).

18. Стойко І. І., Поливода А. В. Машинне навчання та способи його використання. Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій». Тернопіль, 2023. С. 326–327. URL: (https://elartu.tntu.edu.ua/handle/lib/43816) (дата звернення: 08.04.2026).

19. Ліхоткіна В. І. Штучний інтелект як інструмент підтримки управлінських рішень. *Інформаційні технології і системи в документознавчій сфері*. 2025. С. 56–58. URL: (https://jitas.donnu.edu.ua/article/view/17603) (дата звернення: 15.04.2026).

20. П’ятничук І. Інформаційні системи в управлінні проєктами. *Економіка та суспільство*. 2022. URL: (https://economyandsociety.in.ua/index.php/journal/article/view/1618) (дата звернення: 15.04.2026).

21. Davahli M. R. The Last State of Artificial Intelligence in Project Management. 2020. URL: (https://arxiv.org/abs/2012.12262) (дата звернення: 15.04.2026).

22. Dam H. K., Tran T., Grundy J., Ghose A., Kamei Y. Towards Effective AI-Powered Agile Project Management. 2018. URL: (https://arxiv.org/abs/1812.10578) (дата звернення: 15.04.2026).

23. Anghel I., Cioara T. A Systematic Review of Generative AI Usage for IT Project Management. 2026. URL: (https://arxiv.org/abs/2604.21958) (дата звернення: 15.04.2026).

24. Arrieta A. B., Díaz-Rodríguez N., Del Ser J., et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. 2019. URL: (https://arxiv.org/abs/1910.10045) (дата звернення: 24.04.2026).

25. Unterkalmsteiner M., Gorschek T., Islam A. K. M. M., Cheng C. K., Permadi R. B., Feldt R. Evaluation and Measurement of Software Process Improvement: A Systematic Literature Review. 2023. URL: (https://arxiv.org/abs/2307.13143) (дата звернення: 24.04.2026).

ДОДАТОК А. Ключовий фрагмент програмного коду генерації датасету

Лістинг А.1

```
import os
import random
import pandas as pd


DATA_DIR = "data"
DATASET_PATH = os.path.join(DATA_DIR,
"project_tasks_dataset.csv")
RANDOM_SEED = 42


TASK_TEMPLATES = [
    "Implement user authentication with Google OAuth",
    "Create responsive dashboard for analytics",
    "Fix payment processing bug in checkout module",
    "Optimize database queries for financial reports",
    "Add email notifications for new customer orders",
    "Develop admin panel for product management",
    "Improve mobile layout for catalog page",
    "Integrate third-party delivery API",
    "Refactor legacy code in checkout module",
    "Add search and filtering functionality",
    "Implement password reset flow",
    "Develop REST API endpoint for orders",
    "Add role-based access control",
    "Improve application security checks",
    "Optimize image loading performance",
    "Implement unit tests for core services",
    "Add multilingual interface support",
    "Integrate payment provider API",
    "Develop sprint performance dashboard",
    "Implement backlog duplicate detection",
    "Create AI-based task description generator",
]

TEAMS = ["Frontend", "Backend", "QA", "Design", "DevOps",
"Fullstack", "Data", "Mobile"]
PRIORITIES = ["Low", "Medium", "High", "Critical"]
TASK_TYPES = ["Feature", "Bug", "Task", "Improvement",
"Research"]
```

Продовження лістингу А.1

```
COMPONENTS = [
    "Authentication", "Checkout", "Catalog", "Admin Panel",
    "Analytics", "Notifications", "Payments", "Search",
    "API", "Infrastructure", "Mobile UI", "Security",
"Reporting",
]


def calculate_story_points(
    complexity,
    dependencies,
    priority,
    task_type,
    has_external_integration,
    requires_design,
    requires_testing,
    developer_experience,
    backlog_age_days,
):
    """
    Розрахунок Story Points на основі параметрів задачі.
    Логіка імітує спрощене експертне оцінювання в Agile-команді.
    """

    points = complexity

    if dependencies >= 4:
        points += 3
    elif dependencies >= 2:
        points += 2
    elif dependencies == 1:
        points += 1

    if priority == "Critical":
        points += 2
    elif priority == "High":
        points += 1

    if task_type in ["Feature", "Research"]:
        points += 2
    elif task_type in ["Bug", "Improvement"]:
        points += 1

    if has_external_integration == 1:
        points += 2
```

Продовження лістингу А.1

```
    if requires_design == 1:
        points += 1

    if requires_testing == 1:
        points += 1

    if developer_experience <= 2:
        points += 2
    elif developer_experience <= 4:
        points += 1
    elif developer_experience >= 8:
        points -= 1

    if backlog_age_days > 90:
        points += 1

    return max(1, min(points, 13))


def calculate_duration_hours(
    story_points,
    team_size,
    dependencies,
    has_external_integration,
    requires_testing,
    developer_experience,
    sprint_day,
):
    """
    Розрахунок орієнтовної тривалості виконання задачі в
годинах.
    """

    base_hours = story_points * random.uniform(2.5, 4.8)
    dependency_factor = 1 + dependencies * 0.10
    integration_factor = 1.25 if has_external_integration == 1
else 1.0
    testing_factor = 1.15 if requires_testing == 1 else 1.0
    team_factor = max(0.75, 1.20 - team_size * 0.04)
    experience_factor = max(0.75, 1.25 - developer_experience *
0.04)
    sprint_factor = 1.10 if sprint_day >= 10 else 1.0

    duration = (
```

Продовження лістингу А.1

```
            base_hours
            * dependency_factor
            * integration_factor
            * testing_factor
            * team_factor
            * experience_factor
            * sprint_factor
        )

        return round(duration, 1)

    def calculate_delay_probability(
        story_points,
        duration_hours,
        dependencies,
        priority,
        has_external_integration,
        requires_testing,
        developer_experience,
        team_capacity_hours,
    ):
        probability = 0.10

        if story_points >= 8:
            probability += 0.18
        elif story_points >= 5:
            probability += 0.10

        if duration_hours > team_capacity_hours * 0.6:
            probability += 0.20
        elif duration_hours > team_capacity_hours * 0.4:
            probability += 0.10

        if dependencies >= 4:
            probability += 0.18
        elif dependencies >= 2:
            probability += 0.10
        elif dependencies == 1:

            probability += 0.05

        if priority == "Critical":
            probability += 0.10
        elif priority == "High":
            probability += 0.06
```

Продовження лістингу А.1

```
    if has_external_integration == 1:
        probability += 0.12

    if requires_testing == 1:
        probability += 0.06

    if developer_experience <= 2:

        probability += 0.10
    elif developer_experience >= 8:
        probability -= 0.08

    probability += random.uniform(-0.05, 0.05)
    probability = max(0.03, min(probability, 0.95))

    return round(probability, 2)


def calculate_risk_level(delay_probability):
    """
    Перетворення ймовірності затримки у категоріальний рівень
ризику.
    """

    if delay_probability < 0.35:
        return "Low"
    elif delay_probability < 0.65:
        return "Medium"
    else:
        return "High"


def generate_dataset(rows_count=1000):

    random.seed(RANDOM_SEED)
    os.makedirs(DATA_DIR, exist_ok=True)

    rows = []

    for task_id in range(1, rows_count + 1):
        task_description = random.choice(TASK_TEMPLATES)
        team = random.choice(TEAMS)
        priority = random.choice(PRIORITIES)
        task_type = random.choice(TASK_TYPES)
```

Продовження лістингу А.1

```
    component = random.choice(COMPONENTS)

    complexity = random.randint(1, 10)
    dependencies = random.randint(0, 5)
    team_size = random.randint(2, 9)
    sprint_day = random.randint(1, 14)
    backlog_age_days = random.randint(1, 180)
    developer_experience = random.randint(1, 10)

    has_external_integration = random.choice([0, 0, 0, 1])
    requires_design = random.choice([0, 0, 1])
    requires_testing = random.choice([0, 1, 1])

    team_capacity_hours = team_size * 6 * 10

    story_points = calculate_story_points(
        complexity,
        dependencies,
        priority,
        task_type,
        has_external_integration,
        requires_design,
        requires_testing,
        developer_experience,
        backlog_age_days,
    )

    duration_hours = calculate_duration_hours(
        story_points,
        team_size,
        dependencies,
        has_external_integration,
        requires_testing,
        developer_experience,
        sprint_day,
    )

    delay_probability = calculate_delay_probability(
        story_points,
        duration_hours,
        dependencies,
        priority,
        has_external_integration,
        requires_testing,
        developer_experience,
```

Закінчення лістингу А.1

```
            team_capacity_hours,
        )

        risk_level = calculate_risk_level(delay_probability)

        rows.append(
            {
                "task_id": task_id,
                "task_description": task_description,
                "team": team,
                "priority": priority,
                "task_type": task_type,
                "component": component,
                "complexity": complexity,
                "dependencies": dependencies,
                "team_size": team_size,
                "sprint_day": sprint_day,
                "backlog_age_days": backlog_age_days,
                "developer_experience": developer_experience,
                "has_external_integration":
has_external_integration,
                "requires_design": requires_design,
                "requires_testing": requires_testing,
                "team_capacity_hours": team_capacity_hours,
                "story_points": story_points,
                "duration_hours": duration_hours,
                "delay_probability": delay_probability,
                "risk_level": risk_level,
            }
        )

    df = pd.DataFrame(rows)
    df.to_csv(DATASET_PATH, index=False)

    print("Dataset successfully created.")
    print(f"Path: {DATASET_PATH}")
    print(f"Rows count: {len(df)}")


if __name__ == "__main__":
    generate_dataset(rows_count=1000)
```

ДОДАТОК Б. Ключовий фрагмент програмного коду навчання моделей машинного навчання

Лістинг Б.1

```
import os
import joblib
import pandas as pd

from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error,
    r2_score,
    accuracy_score,
    classification_report,
    confusion_matrix,
)
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler


DATASET_PATH = os.path.join("data", "project_tasks_dataset.csv")
MODELS_DIR = "models"

STORY_POINTS_MODEL_PATH = os.path.join(MODELS_DIR,
"story_points_model.pkl")
DURATION_MODEL_PATH = os.path.join(MODELS_DIR,
"duration_model.pkl")
RISK_MODEL_PATH = os.path.join(MODELS_DIR, "risk_model.pkl")

RANDOM_STATE = 42


FEATURE_COLUMNS = [
    "team",
    "priority",
    "task_type",
    "component",
    "complexity",
    "dependencies",
    "team_size",
    "sprint_day",
```

Продовження лістингу Б.1

```
    "backlog_age_days",
    "developer_experience",
    "has_external_integration",
    "requires_design",
    "requires_testing",
    "team_capacity_hours",
]

CATEGORICAL_FEATURES = [
    "team",
    "priority",
    "task_type",
    "component",
]

NUMERIC_FEATURES = [
    "complexity",
    "dependencies",
    "team_size",
    "sprint_day",
    "backlog_age_days",
    "developer_experience",
    "has_external_integration",
    "requires_design",
    "requires_testing",
    "team_capacity_hours",
]


def load_dataset():

    if not os.path.exists(DATASET_PATH):
        raise FileNotFoundError(
            f"Dataset was not found: {DATASET_PATH}. "
            "Run dataset_generator.py before training models."
        )

    return pd.read_csv(DATASET_PATH)


def create_preprocessor():

    categorical_transformer =
OneHotEncoder(handle_unknown="ignore")
    numeric_transformer = StandardScaler()
```

Продовження лістингу Б.1

```
    preprocessor = ColumnTransformer(
        transformers=[
            ("categorical", categorical_transformer,
CATEGORICAL_FEATURES),
            ("numeric", numeric_transformer, NUMERIC_FEATURES),
        ]
    )

    return preprocessor


def train_story_points_model(df):

    x = df[FEATURE_COLUMNS]
    y = df["story_points"]

    x_train, x_test, y_train, y_test = train_test_split(
        x,
        y,
        test_size=0.2,
        random_state=RANDOM_STATE,
    )

    model = Pipeline(
        steps=[
            ("preprocessor", create_preprocessor()),
            (
                "regressor",
                RandomForestRegressor(
                    n_estimators=250,
                    max_depth=14,
                    random_state=RANDOM_STATE,
                ),
            ),
        ]
    )

    model.fit(x_train, y_train)
    predictions = model.predict(x_test)

    metrics = {
        "MAE": mean_absolute_error(y_test, predictions),
        "MSE": mean_squared_error(y_test, predictions),
        "RMSE": mean_squared_error(y_test, predictions) ** 0.5,
```

Продовження лістингу Б.1

```
        "R2": r2_score(y_test, predictions),
    }

    return model, metrics


def train_duration_model(df):

    x = df[FEATURE_COLUMNS]
    y = df["duration_hours"]

    x_train, x_test, y_train, y_test = train_test_split(
        x,
        y,
        test_size=0.2,
        random_state=RANDOM_STATE,
    )

    model = Pipeline(
        steps=[
            ("preprocessor", create_preprocessor()),
            (
                "regressor",
                RandomForestRegressor(
                    n_estimators=300,
                    max_depth=16,
                    random_state=RANDOM_STATE,
                ),
            ),
        ]
    )

    model.fit(x_train, y_train)
    predictions = model.predict(x_test)

    metrics = {
        "MAE": mean_absolute_error(y_test, predictions),
        "MSE": mean_squared_error(y_test, predictions),
        "RMSE": mean_squared_error(y_test, predictions) ** 0.5,
        "R2": r2_score(y_test, predictions),
    }

    return model, metrics
```

Продовження лістингу Б.1

```
def train_risk_model(df):
    x = df[FEATURE_COLUMNS]
    y = df["risk_level"]

    x_train, x_test, y_train, y_test = train_test_split(
        x,
        y,
        test_size=0.2,
        random_state=RANDOM_STATE,
        stratify=y,
    )

    model = Pipeline(
        steps=[
            ("preprocessor", create_preprocessor()),
            (
                "classifier",
                RandomForestClassifier(
                    n_estimators=250,
                    max_depth=14,
                    class_weight="balanced",
                    random_state=RANDOM_STATE,
                ),
            ),
        ]
    )

    model.fit(x_train, y_train)
    predictions = model.predict(x_test)

    metrics = {
        "Accuracy": accuracy_score(y_test, predictions),
        "Classification Report": classification_report(y_test,
predictions),
        "Confusion Matrix": confusion_matrix(y_test,
predictions),
    }

    return model, metrics

def main():

    os.makedirs(MODELS_DIR, exist_ok=True)
```

Закінчення лістингу Б.1

```
    df = load_dataset()

    story_points_model, story_metrics =
train_story_points_model(df)
    duration_model, duration_metrics = train_duration_model(df)
    risk_model, risk_metrics = train_risk_model(df)

    joblib.dump(story_points_model, STORY_POINTS_MODEL_PATH)
    joblib.dump(duration_model, DURATION_MODEL_PATH)
    joblib.dump(risk_model, RISK_MODEL_PATH)

    print("MODEL TRAINING RESULTS")
    print("Story Points Regression Model:")
    print(f"MAE:  {story_metrics['MAE']:.4f}")
    print(f"RMSE: {story_metrics['RMSE']:.4f}")
    print(f"R2:   {story_metrics['R2']:.4f}")

    print("\nDuration Hours Regression Model:")
    print(f"MAE:  {duration_metrics['MAE']:.4f}")
    print(f"RMSE: {duration_metrics['RMSE']:.4f}")
    print(f"R2:   {duration_metrics['R2']:.4f}")

    print("\nRisk Level Classification Model:")
    print(f"Accuracy: {risk_metrics['Accuracy']:.4f}")
    print("Classification Report:")
    print(risk_metrics["Classification Report"])
    print("Confusion Matrix:")
    print(risk_metrics["Confusion Matrix"])

    print("\nModels were successfully saved:")
    print(f"- {STORY_POINTS_MODEL_PATH}")
    print(f"- {DURATION_MODEL_PATH}")
    print(f"- {RISK_MODEL_PATH}")


if __name__ == "__main__":
    main()
```

ДОДАТОК В. Ключовий фрагмент програмного коду користувацького інтерфейсу AI-асистента

Лістинг В.1

```
import os
import joblib
import pandas as pd
import streamlit as st


MODELS_DIR = "models"

STORY_POINTS_MODEL_PATH = os.path.join(MODELS_DIR,
"story_points_model.pkl")
DURATION_MODEL_PATH = os.path.join(MODELS_DIR,
"duration_model.pkl")
RISK_MODEL_PATH = os.path.join(MODELS_DIR, "risk_model.pkl")


FEATURE_COLUMNS = [
    "team",
    "priority",
    "task_type",
    "component",
    "complexity",
    "dependencies",
    "team_size",
    "sprint_day",
    "backlog_age_days",
    "developer_experience",
    "has_external_integration",
    "requires_design",
    "requires_testing",
    "team_capacity_hours",
]


@st.cache_resource
def load_models():
    story_points_model = joblib.load(STORY_POINTS_MODEL_PATH)
    duration_model = joblib.load(DURATION_MODEL_PATH)
    risk_model = joblib.load(RISK_MODEL_PATH)

    return story_points_model, duration_model, risk_model
```

Продовження лістингу В.1

```
def detect_task_type(task_text):
    text = task_text.lower()

    if any(word in text for word in ["fix", "bug", "error",
"виправити", "помилка"]):
        return "Bug"

    if any(word in text for word in ["improve", "optimize",
"покращити", "оптимізувати"]):
        return "Improvement"

    if any(word in text for word in ["add", "create", "develop",
"implement", "додати", "створити", "реалізувати"]):
        return "Feature"

    return "Task"


def detect_component(task_text):
    text = task_text.lower()

    if any(word in text for word in ["oauth", "google", "login",
"auth", "авторизація"]):
        return "Authentication"

    if any(word in text for word in ["payment", "checkout",
"order", "оплата", "замовлення"]):
        return "Checkout"

    if any(word in text for word in ["catalog", "product",
"filter", "каталог", "товар"]):
        return "Catalog"

    if any(word in text for word in ["api", "integration",
"інтеграція"]):
        return "API"

    return "API"


def create_input_dataframe(
    team,
    priority,
    task_type,
```

Продовження лістингу В.1

```
    component,
    complexity,
    dependencies,
    team_size,
    sprint_day,
    backlog_age_days,
    developer_experience,
    has_external_integration,
    requires_design,
    requires_testing,
):
    team_capacity_hours = team_size * 6 * 10

    input_data = {
        "team": team,
        "priority": priority,
        "task_type": task_type,
        "component": component,
        "complexity": complexity,
        "dependencies": dependencies,
        "team_size": team_size,
        "sprint_day": sprint_day,
        "backlog_age_days": backlog_age_days,
        "developer_experience": developer_experience,
        "has_external_integration": has_external_integration,
        "requires_design": requires_design,
        "requires_testing": requires_testing,
        "team_capacity_hours": team_capacity_hours,
    }

    return pd.DataFrame([input_data], columns=FEATURE_COLUMNS)


def generate_user_story(component):
    return (
        f"Як користувач, я хочу отримати функціональність, "
        f"пов’язану з модулем {component}, щоб ефективніше
працювати з продуктом."
    )


def generate_acceptance_criteria(task_type):
    criteria = [
        "Функціональність реалізована відповідно до опису
задачі.",
```

Продовження лістингу В.1

```
            "Основний користувацький сценарій працює коректно.",
            "Задача пройшла перевірку та готова до QA-валідації.",
        ]

        if task_type == "Bug":
            criteria.append("Помилка більше не відтворюється після
внесення змін.")

        return criteria


def generate_recommendation(risk_level):
    if risk_level == "High":
        return "Рекомендується переглянути задачу, уточнити
залежності та розділити її на менші підзадачі."

    if risk_level == "Medium":
        return "Задачу можна включати до спринту, але потрібно
контролювати ризики та критерії приймання."

    return "Задача має низький ризик і може бути запланована в
межах стандартного процесу."


def main():
    st.title("AI-асистент для Project та Product Management")

    story_points_model, duration_model, risk_model =
load_models()

    task_text = st.text_area(
        "Короткий опис задачі",
        "Реалізувати авторизацію через Google OAuth для
мобільного застосунку",
    )

    team = st.selectbox("Команда", ["Frontend", "Backend", "QA",
"Design", "DevOps", "Mobile"])
    priority = st.selectbox("Пріоритет", ["Low", "Medium",
"High", "Critical"])

    task_type = detect_task_type(task_text)
    component = detect_component(task_text)

    complexity = st.slider("Складність задачі", 1, 10, 6)
```

Продовження лістингу В.1

```
    dependencies = st.slider("Кількість залежностей", 0, 5, 2)
    team_size = st.slider("Розмір команди", 2, 9, 5)
    sprint_day = st.slider("День спринту", 1, 14, 4)
    backlog_age_days = st.slider("Вік задачі у беклозі", 1, 180,
30)
    developer_experience = st.slider("Досвід розробника", 1, 10,
6)

    has_external_integration = st.checkbox("Потребує зовнішньої
інтеграції", value=True)
    requires_design = st.checkbox("Потребує UI/UX-дизайну",
value=False)
    requires_testing = st.checkbox("Потребує тестування",
value=True)

    if st.button("Згенерувати та спрогнозувати"):
        input_df = create_input_dataframe(
            team,
            priority,
            task_type,
            component,
            complexity,
            dependencies,
            team_size,
            sprint_day,
            backlog_age_days,
            developer_experience,
            int(has_external_integration),
            int(requires_design),
            int(requires_testing),
        )

        predicted_story_points =
round(float(story_points_model.predict(input_df)[0]))
        predicted_duration =
round(float(duration_model.predict(input_df)[0]), 1)
        predicted_risk = risk_model.predict(input_df)[0]

        user_story = generate_user_story(component)
        acceptance_criteria =
generate_acceptance_criteria(task_type)
        recommendation = generate_recommendation(predicted_risk)

        st.subheader("Результат прогнозування")
        st.write(f"Story Points: {predicted_story_points}")
```

Закінчення лістингу В.1

```
        st.write(f"Орієнтовна тривалість: {predicted_duration}
год")
        st.write(f"Рівень ризику: {predicted_risk}")

        st.subheader("User Story")
        st.write(user_story)

        st.subheader("Acceptance Criteria")
        for index, criterion in enumerate(acceptance_criteria,
start=1):
            st.write(f"{index}. {criterion}")

        st.subheader("Рекомендація менеджеру")
        st.write(recommendation)


if __name__ == "__main__":
    main()
```