МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 124 «Системний аналіз»
Освітня програма «Інформаційні технології інтелектуального аналізу даних та проєктування програмних систем»

«Допущений до захисту»
Завідувач кафедри _____
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Математична модель для прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, на ранніх етапах розробки та її програмна реалізація.

Виконав: студент групи 6153м
_____ Смітієнко В. О.
(підпис, ПІБ)

Керівник роботи:
Доцент кафедри ПЗАС, к.т.н., доц.
Каіров В. О.
(посада, науковий ступень, вчене звання)
_____
(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 124 «Системний аналіз»
Освітня програма «Інформаційні технології інтелектуального аналізу даних та проєктування програмних систем»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____доц. Л.О. Латанська
(підпис)
«14» жовтня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «магістр»

Студенту Смітієнко Владиславу Олексійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Математична модель для прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, на ранніх етапах розробки та її програмна реалізація».

Керівник роботи Каіров В. О.

Затверджені наказом ректора № 1222-УЧ уч від «15» _____10_____ 2025 р.

2. Термін подання роботи: 08.12.2025 р. _____

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) _____

Вступ (Актуальність теми. Зв'язок роботи з науковими програмами, планами, темами. Мета і завдання дослідження. Об'єкт дослідження. Предмет дослідження. Методи дослідження. Наукова новизна одержаних результатів. Практичне значення одержаних результатів. Особистий внесок здобувача. Апробація результатів досліджень**.** Публікації.); Огляд літератури та обґрунтування необхідності проведення досліджень за обраною темою; Викладення результатів власних досліджень з висвітленням того нового, що пропонується; Розділ з розробки проєктних рішень для програмної системи; Організаційно-економічний розділ; Розділи з охорони праці та охорони навколишнього середовища; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма і методика випробувань програмної системи)

5. Перелік презентаційних матеріалів 1.Тема, 2. Мета та завдання дослідження, 3. Об'єкт та предмет дослідження, 4. Методи дослідження, 5. Наукова новизна одержаних результатів, 6. Практичне значення одержаних результатів, 7. Особистий внесок здобувача 8. Апробація результатів досліджень та публікації, 9. Аналіз сучасних методів прогнозування ПС. 10. Обґрунтування необхідності проведення досліджень за обраною темою 11. Пошук та аналіз емпіричних даних, 12. Нормалізація даних та вилучення викидів, 13. Нелінійна регресійна модель, 14. Порівняння побудованої моделі з існуючими, 15. Діаграма варіантів використання системи, 16. Діаграма діяльності системи, 17. Діаграма класів системи, 18. Діаграма послідовності системи, 19. Інтерфейс користувача системи. 20. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 27.10.2025 р. _____

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	16.10.2025	НС*
2.	з 01.09.2025 по 26.10.2025 р.	18.10.2025	НС*
3.	Підготовка розділу (ів) за результатами власних досліджень	21.10.2025	НС*
4.	Підготовка розділу з розробки проєктних рішень для програмної системи	27.11.2025	
5.	Підготовка організаційно-економічного розділу	29.11.2025	
6.	Підготовка розділу з охорони праці	02.12.2025	
7.	Підготовка розділу з охорони навколишнього середовища	04.12.2025	
8.	Підготовка розділу ВИСНОВКИ	05.12.2025	
9.	Оформлення списку використаних джерел та додатків	06.12.2025	
10.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версії роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	07.12.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	08.12.2025	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	15.12.2025	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файла-опису кваліфікаційної роботи (згідно з Додатком до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	19.12.2025	

* - за результатами дослідницької практики (ДП), яка була з 01.09.2025 по 26.10.2025 р.

Студент _____ Смітієнко В. О.
(підпис) (ПІБ)

Керівник роботи _____ Каіров В. О.
(підпис) (ПІБ)

РЕФЕРАТ

Смітієнко Владислав Олексійович

«Математична модель для прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, на ранніх етапах розробки та її програмна реалізація»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 124 «Системний аналіз» освітньої програми «Інформаційні технології інтелектуального аналізу даних та проєктування програмних систем». Національний університет кораблебудування імені адмірала. Миколаїв, 2025 р.

Обсяг роботи: 86 стор., 9 табл., 6 рис., 33 використаних джерел, 5 додатків

Актуальність теми: полягає в необхідності більш достовірного прогнозування розміру програмних систем (ПС), розроблених з використанням фреймворку Yii, на ранніх етапах розробки через низьку достовірність існуючих математичних моделей.

Мета дослідження. Мета роботи полягає у підвищенні достовірності прогнозування розміру ПС, розроблених з використанням фреймворку Yii, за допомогою математичних моделей.

Об’єкт дослідження: процес прогнозування розміру ПС, розроблених з використанням фреймворку Yii.

Предмет дослідження: математичні моделі для прогнозування розміру ПС, розроблених з використанням фреймворку Yii.

Методи дослідження: математична статистика та аналіз даних, регресійний аналіз, чисельні методи та оптимізація, ймовірнісний аналіз, об'єктно-орієнтоване програмування.

Наукова новизна одержаних результатів: удосконалено математичну модель для раннього прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, шляхом побудови нелінійної регресійної моделі з багатовимірним нормалізуючим перетворенням Бокса-Кокса. Це забезпечує

підвищення достовірності раннього прогнозування розміру у порівнянні з існуючими моделями для PHP фреймворків CakePHP та CodeIgniter.

Практичне значення одержаних результатів. Результатом роботи є програмна система для використання спеціалізованої нелінійної регресійної моделі, призначеної для достовірного оцінювання розміру ПС, розроблених з використанням фреймворку Yii. Це дозволяє керівникам проєктів та архітекторам отримувати обґрунтовані прогнози обсягу коду вже на етапі проєктування архітектури, що значно підвищує якість планування ресурсів, бюджету та термінів розробки для нових проєктів, особливо в контексті переходу на версію Yii 3.0.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи оприлюдненні на VI Міжнародній науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (м. Миколаїв, 16-17 листопада 2025 року).

Публікації: Основні положення й результати кваліфікаційної роботи опубліковано у 1 науковій праці – тезах конференції.

Ключові слова: математичні моделі, прогнозування розміру, LOC, програмні системи, PHP, фреймворк, нелінійні регресійні моделі.

ABSTRACT

Smitiienko Vladyslav Oleksiiovych

«Mathematical model for predicting the size of software systems, developed using the Yii framework, at early stages of development and its software implementation»

Qualification work for obtaining a master's degree in specialty 124 – "System Analysis". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2025.

Volume: 86 p., 9 tables, 6 figures, 33 references, 5 appendices.

Topic Relevance: lies in the need for more reliable early-stage prediction of the size of software systems (SS) developed using the Yii framework, due to the low accuracy of existing mathematical models.

Research goal. The goal of the work is to increase the reliability of predicting the size of software systems developed using the Yii framework by means of mathematical models.

Object of research: the process of early size estimation (in lines of code) for software systems developed using the Yii framework.

Subject of research: mathematical models and methods for estimating the size of software systems developed using the Yii framework.

Methods of research: mathematical statistics and data analysis, regression analysis, numerical methods and optimization, probabilistic analysis, object-oriented programming.

Scientific contribution: an improved mathematical model for early prediction of the size of software systems developed using the Yii framework has been proposed by constructing a nonlinear regression model with a multivariate normalizing Box–Cox transformation. This ensures higher reliability of early size prediction compared to existing models for the PHP frameworks CakePHP and CodeIgniter.

Practical value of obtained results. The result of the work is the development of the «YiiSizeEstimator» tool, which implements the specialized nonlinear regression model for reliable estimation of the size of Yii-based systems. This enables project managers and architects to obtain substantiated estimates of code volume already at the

architectural design stage, significantly improving the quality of resource, budget, and timeline planning for new projects, especially in the context of the transition to Yii 3.0.

Approbation of the thesis results. The main provisions and results of the qualification work were presented at the VI International Scientific and Practical Internet Conference «Information Technologies: Models, Algorithms, Systems – 2025» (Mykolaiv, November 16-17, 2025).

Publications. The main provisions and results of the qualification work are published in 1 scientific work – conference proceedings.

Keywords: mathematical models, size prediction, LOC, software systems, PHP, Yii framework, nonlinear regression models, Box-Cox transformation.

ЗМІСТ

ВСТУП

Актуальність теми. Достовірне прогнозування розміру програмних систем (ПС) є критичним етапом управління проєктами, безпосередньо впливаючи на прогнозування витрат, зусиль та необхідних ресурсів [1]. Основним інструментом для такого прогнозування є параметричні моделі, такі як COCOMO II, які базуються на метриках розміру [2].

Фреймворк Yii посідає значне місце в розробці корпоративних веб-додатків завдяки своїй структурі, безпеці та підтримці патерну MVC [3]. Його популярність очікувано зросте з виходом мажорної версії Yii 3.0, що актуалізує потребу в точних інструментах планування для нових проєктів.

На практиці існує широкий спектр методів прогнозування – від оцінювання рядків коду (LOC) [4 - 11] до аналізу функціональних точок (FPA) [12]. Дослідження в галузі прогнозування розміру ПЗ вказують на суттєву залежність достовірності моделей від технологічного контексту [13, 14]. Попередні роботи демонструють, що універсальні LOC-моделі або моделі, калібровані для одних фреймворків, можуть давати похибку прогнозування розміру при застосуванні до проєктів, розроблених на інших платформах. Це пов’язано з тим, що кожен сучасний фреймворк, включно з Yii, має унікальні архітектурні патерни, ідіоми коду та рівень абстракції, що безпосередньо впливає на співвідношення між програмними метриками та фінальним обсягом коду. Це створює потребу в розробці спеціалізованих моделей, адаптованих до конкретних фреймворків або типу проєктів.

Для побудови адекватної прогнозної моделі в даній роботі обрано багатовимірне перетворення Бокса-Кокса. Цей вибір зумовлений тим, що реальні дані проєктів ПЗ часто суттєво відхиляються від нормального розподілу, що робить пряме застосування класичної лінійної регресії неможливим або малоефективним. Зазначене перетворення є ключовим інструментом для підготовки даних до побудови саме нелінійних регресійних моделей, оскільки дозволяє ефективно

нормалізувати розподіл, врахувати кореляційну структуру між метриками та забезпечити виконання статистичних припущень, необхідних для отримання стійких і точних оцінок. Підхід, заснований на застосуванні нормалізуючих перетворень, широко застосовується у задачі прогнозування розміру ПЗ.

Метою даної роботи є підвищення достовірності прогнозування розміру ПС, розроблених з використанням фреймворку, Yii за допомогою математичних моделей.

Для досягнення поставленої мети сформульовано наступні завдання:

– Провести аналіз сучасних методів прогнозування розміру ПС з акцентом на підходи, засновані на метриках коду.

– Сформувати емпіричний набір даних шляхом збору метрик реальних проєктів на Yii.

– Побудувати математичну модель, застосовуючи багатовимірні нормалізуючі перетворення для підготовки даних.

– Розробити програмну систему для автоматизації процесу прогнозування на основі створеної моделі.

Об’єкт дослідження: процес прогнозування розміру ПС, розроблених з використанням фреймворку Yii.

Предмет дослідження: нелінійні регресійні моделі прогнозування розміру ПС, розроблених з використанням фреймворку Yii

Методи дослідження: математична статистика та аналіз даних, регресійний аналіз, чисельні методи та оптимізація, ймовірнісний аналіз, об'єктно-орієнтоване програмування.

Наукова новизна одержаних результатів: удосконалено математичну модель для раннього прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, шляхом побудови нелінійної регресійної моделі з багатовимірним нормалізуючим перетворенням Бокса-Кокса. Це забезпечує підвищення достовірності раннього прогнозування розміру у порівнянні з існуючими моделями для PHP фреймворків CakePHP та CodeIgniter.

Практичне значення одержаних результатів. Результатом роботи є програмна система для використання спеціалізованої нелінійної регресійної моделі, призначеної для достовірного оцінювання розміру ПС, розроблених з використанням фреймворку Yii. Це дозволяє керівникам проєктів та архітекторам отримувати обґрунтовані прогнози обсягу коду вже на етапі проєктування архітектури, що значно підвищує якість планування ресурсів, бюджету та термінів розробки для нових проєктів, особливо в контексті переходу на версію Yii 3.0.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною науковою працею. Усі наукові результати отримано автором особисто. У праці [15], яка була опублікована у співавторстві, автору належить нелінійна регресійна модель для прогнозування розміру ПС, розроблених з використанням фреймворку Yii, на основі багатовимірного нормалізуючого перетворення Бокса-Кокса.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи опубліковані на VI Міжнародній науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи – 2025» (м. Миколаїв, 16-17 листопада 2025 року).

Публікації: Основні положення й результати кваліфікаційної роботи опубліковано у 1 науковій праці – тезах конференції.

Ключові слова: математичні моделі, прогнозування розміру, LOC, програмні системи, PHP, фреймворк, нелінійні регресійні моделі.

1 АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПРОГНОЗУВАННЯ РОЗМІРУ ПС ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ДОСЛІДЖЕННЯ

1.1 Аналіз існуючих методів та математичних моделей для прогнозування розміру ПС

У сучасному стрімкому середовищі розробки ПС точне прогнозування розміру проєкту є критичним фактором успіху, оскільки саме воно лежить в основі ефективного планування [1]. Визначення масштабів майбутнього проєкту дозволяє прогнозувати необхідні ресурси, час та вартість, забезпечуючи його стабільний старт та управління на всьому життєвому циклі. По суті, цей процес є систематичним визначенням обсягу робіт і ресурсів, необхідних для успішного завершення. Він передбачає комплексну оцінку різних аспектів, таких як функціональні вимоги та технічна складність, з метою прогнозування зусиль, тривалості та вартості, формуючи тим самим обґрунтовану основу для детального планування та контролю.

Важливість достовірного прогнозування проявляється в кількох ключових аспектах управління проєктами. По-перше, воно є основою для реалістичного фінансового планування, допомагаючи сформувати бюджет і уникнути дефіциту коштів. По-друге, воно забезпечує правильне планування та виділення людських, технічних і матеріальних ресурсів. Крім того, прогнозування сприяє створенню досяжних графіків робіт і контрольних точок, а також допомагає на ранній стадії виявити потенційні ризики для виконання проєкту. Всі ці складові разом закладають фундамент для детального планування всього процесу реалізації та дозволяють заздалегідь інтегрувати заходи щодо забезпечення якості.

Оскільки прогнозування – це багатогранне завдання, воно вимагає участі цілої команди фахівців. Керівник проєкту відповідає за загальну організацію цього процесу. Експерти з предметної області та бізнес-аналітики забезпечують глибоке розуміння бізнес-контексту та формують чіткі вимоги, які є первинною основою

для будь-яких розрахунків. З технічної сторони, технічні ліди та розробники оцінюють складність архітектури, розробки та інтеграції. Фінансові аналітики прогнозують витрати, а менеджери з ризиків оцінюють потенційні загрози для обсягу робіт та термінів. Важливу роль також відіграють замовники, які забезпечують вхідні дані щодо очікувань та пріоритетів.

У світі управління проєктами розробки ПС існує цілий спектр методів прогнозування, кожен з яких має свою специфіку та оптимальну область застосування.

Одним з найпоширеніших є метод експертної оцінки (Expert Judgment) [16], який спирається на колективний досвід та інтуїцію кваліфікованих фахівців. Цей підхід часто використовують на ранніх етапах, коли інформації про проєкт ще обмаль, але необхідно отримати швидку приблизну оцінку.

Коли в арсеналі організації є історія реалізованих ініціатив, на допомогу приходить аналогічне оцінювання (Analogous Estimation) [17]. Його суть полягає у пошуку та аналізі минулих проєктів, схожих за масштабом та складністю, і екстраполяції їх даних на поточне завдання. Цей метод відносно швидкий, але його точність залежить від якості історичної бази даних та реальної схожості проєктів.

Для більш детального та точного планування застосовують оцінювання «знизу вгору» (Bottom-up Estimation) [18]. Проєкт розбивається на мінімальні, легко оцінювані компоненти або задачі, для кожного з яких визначається трудомісткість. Підсумовуючи ці детальні оцінки, отримують загальну картину. Незважаючи на високу точність, цей метод вимагає значно більше часу та чітко визначеної структури робіт.

Щоб врахувати невизначеність та ризики, використовують триточкове оцінювання (Three-point Estimation). Замість однієї цифри фахівці оцінюють три сценарії: оптимістичний, песимістичний та найбільш імовірний. За допомогою формул (наприклад, PERT) ці значення об'єднуються в єдину очікувану оцінку, що дає розуміння можливого діапазону результатів [19].

Серед алгоритмічних методів особливе місце займає аналіз функціональних точок (Function Points). Цей підхід абстрагується від технологій і оцінює розмір

системи на основі її функціональності для користувача, враховуючи такі елементи, як вхідні/вихідні дані, запити та файли. Це дозволяє отримати об'єктивну метрику на ранніх етапах, коли технічні деталі ще невідомі [12].

Схожий за філософією, але орієнтований на об'єктно-орієнтований аналіз метод – оцінювання за точками прецедентів (Use Case Points). Він вимірює складність через призму кількості та складності сценаріїв використання (use cases) і кількості акторів. Цей метод добре вписується в процеси, засновані на UML [20].

Більш формалізованим є параметричне оцінювання (Parametric Estimation), яке використовує математичні моделі, що зв'язують ключові параметри проєкту (наприклад, кількість рядків коду або функціональних точок) з трудомісткістю та термінами, виводячи розрахунки на основі статистики попередніх проєктів.

Найвідомішою реалізацією такого підходу є конструктивна модель вартості COCOMO (Constructive Cost Model) [2]. Ця алгоритмічна модель враховує численні фактори, такі як розмір продукту, атрибути персоналу та проєкту, апаратного забезпечення, для розрахунку зусиль, тривалості та вартості розробки.

Щоб мінімізувати групову упередженість та досягти консенсусу в експертних оцінках, застосовують метод широкого дельфі (Wideband Delphi) [21]. Він поєднує анонімність індивідуальних оцінок фахівців з кількома ітеративними раундами обговорення, що дозволяє поступово зблизити думки та отримати збалансований, узгоджений результат.

Незважаючи на наявність структурованих методів, фахівці стикаються з низкою викликів. Основні проблеми включають невизначеність та мінливість вимог, суб'єктивність експертних оцінок, брак якісних історичних даних для порівняння, а також складність оцінки нетехнічних робіт. Для підвищення достовірності рекомендується використовувати комбінацію кількох методів, постійно накопичувати та аналізувати метрики з завершених проєктів, застосовувати ітеративний підхід із регулярним переглядом оцінок, а також культивувати відкриту культуру, де реалістичні оцінки мають пріоритет над надмірно оптимістичними.

1.2 Застосування LOC-прогнозування для визначення розміру ПС

На відміну від експертних методів (як Wideband Delphi або експертна оцінка), які значною мірою суб’єктивні, LOC є чіткою кількісною мірою. Ця кількісна природа дозволяє встановлювати статистично значимі кореляції між обсягом коду та такими параметрами, як:

– Трудомісткість розробки (людино-місяці).

– Кількість дефектів.

– Складність підтримки.

Це дає змогу розробляти прогнозні регресійні моделі. Наприклад, модель виду $Effort = a * KLOC^b + c$, де a, b, c – коефіцієнти моделі, а KLOC – прогнозовані тисячі рядків коду. Такі моделі, подібні до COCOMO та COCOMO II, але часто більш специфічні для конкретної команди або технологічного стеку, дають об'єктивну, числову оцінку, вільну від групових упереджень.

Одна з найважливіших практичних переваг – можливість застосування моделей, заснованих на LOC- прогнозуванні, вже на етапі архітектурного та детального проєктування, коли реального коду ще немає. Ключем тут є використання метрик об'єктно-орієнтованого дизайну, таких як метрики з набору CK (Chidamber & Kemerer Suite), які можна розрахувати безпосередньо з UML-діаграм класів [22]:

– WMC (Weighted Methods per Class) – Сумарна складність методів класу. Прямо корелює з обсягом коду, який буде реалізовано в класі.

– DIT (Depth of Inheritance Tree) – Глибина дерева успадкування.

– NOC (Number of Children) – Кількість нащадків.

– CBO (Coupling Between Objects) – Зв'язаність між об'єктами.

– RFC (Response For a Class) – Кількість методів, які можуть бути викликані у відповідь на повідомлення класу.

– LCOM (Lack of Cohesion in Methods) – Відсутність зв'язності методів.

Метрики CK, що в тому числі можуть використовуються для прогнозування LOC, слугують не лише вхідними даними для такої математичної моделі, а й

потужними індикаторами якості дизайну, що дозволяє вирішувати дві ключові задачі одночасно: прогнозування зусиль та проактивне управління складністю. Таким чином, модель оцінювання автоматично ідентифікує компоненти, що потребують архітектурного рефакторингу ще до початку кодування, перетворюючи інструмент планування на ефективний механізм превентивного контролю якості.

Дослідження [4 - 11] для різних мов програмування (зокрема для PHP фреймворків) показують, що між цими метриками та кінцевим обсягом коду (LOC) існують стабільні статистичні залежності, саме тому проаналізувавши діаграми класів проєкту, можна:

1) Визначити метрики.

2) За допомогою попередньо навченої регресійної моделі виконати прогнозування LOC для кожного класу та системи в цілому.

3) Підставити отриману оцінку LOC у калібровану модель витрат (наприклад, у COCOMO або COCOMO II).

1.3 Застосування нормалізуючих перетворень при LOC-прогнозуванні

Застосування статистичних методів, підходів та моделей для LOC-оцінювання, особливо при використанні емпіричних даних з метрик (наприклад, CK Suite), часто вимагає попередньої нормалізації вихідних даних. Ця необхідність випливає з властивостей реальних проєктних даних, які зазвичай мають нерівномірний (ненормальний) розподіл: вони можуть бути сильно асиметричними (скошеними), містити викиди (outliers) або демонструвати гетероскедастичність (непостійність дисперсії). Однак багато поширених статистичних методів (наприклад, лінійна регресія, яка часто лежить в основі прогнозних моделей) та критеріїв перевірки гіпотез передбачають, що залишки моделі розподілені нормально або що самі дані наближаються до нормального розподілу. Недотримання цієї умови може призвести до значного зниження достовірності прогнозів та вірогідності статистичних висновків. Для вирішення цієї проблеми в

практиці програмної інженерії активно застосовуються різні нормалізуючі перетворення.

Одним з найпростіших і найпоширеніших методів є логарифмічне перетворення (log10 або натуральний логарифм ln). Воно ефективно зменшує позитивну асиметрію даних, «стискаючи» екстремальні значення та наближаючи розподіл до нормального. Це перетворення особливо корисне для даних про розмір ПЗ, які часто мають розподіл, близький до логнормального.

Більш гнучким і потужним підходом є перетворення Бокса-Кокса (Box-Cox). Воно представляє собою сімейство степеневих перетворень, параметр якого (λ) підбирається автоматично на основі даних таким чином, щоб максимізувати правдоподібність отриманого розподілу щодо нормального. Стандартне одновимірне перетворення Бокса-Кокса застосовується до кожної змінної окремо. Однак у випадку LOC-прогнозування ми часто маємо справу з багатовимірними даними (набір взаємопов'язаних метрик). Для таких випадків існує множинне (multivariate) перетворення Бокса-Кокса, яке враховує кореляційну структуру між змінними та знаходить оптимальні параметри трансформації для всієї множини одночасно, зберігаючи між ними статистичні взаємозв'язки, що критично важливо для подальшого моделювання.

Альтернативою слугує перетворення Джонсона (Johnson normalizing transformation), яке, аналогічно Бокса-Кокса, прагне знайти найкращу трансформацію для приведення даних до нормального розподілу. Воно охоплює три сімейства перетворень (SB, SL, SU), що дозволяє ефективно працювати з різними типами ненормальних розподілів, включаючи обмежені або необмежені дані.

Використання цих перетворень є важливим етапом підготовки даних перед побудовою прогнозної моделі. Вони підвищують надійність та стабільність статистичних оцінок, дозволяють коректно застосовувати параметричні методи та, як наслідок, підвищують точність та обґрунтованість кінцевих прогнозів зусиль та розміру на основі LOC та метрик дизайну.

1.4 Обґрунтування необхідності проведення досліджень

Проведений аналіз існуючих підходів демонструє, що незважаючи на широкий вибір методів прогнозування розміру, LOC-прогнозування на основі метрик дизайну має значний потенціал для отримання об’єктивних ранніх оцінок розміру ПЗ [4 - 11]. Широковідомий метод функціональних точок (Function Points) довів свою ефективність як міра бізнес-функціональності та застосовується для високорівневої оцінки [12]. Однак, будучи технологічно незалежним, цей підхід часто не враховує специфіку конкретного фреймворку або типу проєкту. Для архітектурно-орієнтованих фреймворків, таких як Yii, де певні паттерни та рівні абстракції можуть значно впливати на співвідношення між функціональністю та обсягом коду, універсальні оцінки на основі функціональних точок можуть давати значні відхилення.

Існуючі дані свідчать, що застосування універсальних або побудованих для інших технологічних стеків моделей (наприклад, для інших фреймворків мови PHP) дає відчутно гірші результати в порівнянні з моделями, створеними під конкретний фреймворк [13, 14]. Це обумовлено унікальними архітектурними патернами, ідіомами коду та рівнем абстракції, властивими кожному фреймворку, що безпосередньо впливає на зв'язок між метриками дизайну та фінальним обсягом коду.

Фреймворк Yii широко використовується для розробки корпоративних веб-додатків через свою архітектурну строгість, безпеку та підтримку патерну MVC [3]. Актуальність побудови такої математичної моделі саме зараз є особливо високою, оскільки наближається реліз Yii 3.0 (готовність оцінюється в 97% на офіційному сайті) [23]. Для цього фреймворку відома [24] нелінійна регресійна модель для прогнозування розміру ПС, розроблених з використанням фреймворку Yii з застосування багатовимірного нормалізуючого перетворення Бокса-Кокса, втім з моменту побудови цієї моделі кодова база проектів могла зазнати суттєвих змін. Це в свою чергу може негативно відобразитись на здатності існуючої моделі прогнозувати розмір. Саме тому наявність спеціалізованого інструменту для

точного оцінювання напередодні цього етапу стає критично важливою для ефективного планування майбутніх проєктів. Вихід мажорної, оновленої версії фреймворку з високою ймовірністю викличе нову хвилю інтересу та активного застосування його в промислових проєктах. Саме тому наявність спеціалізованого інструменту для точного оцінювання напередодні цього етапу стає критично важливою для ефективного планування майбутніх проєктів.

Ключовою проблемою, яку вирішує це дослідження, є низька достовірність класичних лінійних регресійних моделей для опису зв’язку між метриками складності та розміром коду в контексті Yii.

Реальні дані проєктів програмного забезпечення часто демонструють нелінійні та залежні від масштабу взаємозв’язки. Тому основним завданням цієї роботи є побудова саме нелінійної регресійної моделі, здатної краще відображати таку природу залежностей емпіричних даних з метрик проєктів фреймворку Yii.

1.5 Висновки до розділу 1

У розділі 1 проведено аналіз існуючих підходів до прогнозування розміру ПС. Систематизовано класичні методи, такі як експертна оцінка, аналогічне оцінювання, функціональні точки та параметричні моделі (COCOMO), визначено їх переваги та обмеження. Значну увагу приділено LOC-орієнтованому підходу, що базується на метриках об’єктно-орієнтованого дизайну, і обґрунтовано його ефективність для раннього прогнозування на етапі проєктування. Крім того, розглянуто необхідність застосування нормалізуючих перетворень (Box-Cox, Johnson) для підвищення точності статистичних моделей. Остаточно обґрунтована потреба у створенні спеціалізованої нелінійної регресійної моделі для фреймворку Yii, що враховує його архітектурні особливості та актуальність у зв’язку з виходом версії 3.0.

2 МАТЕМАТИЧНІ МОДЕЛЬ ДЛЯ ПРОГНОЗУВАННЯ РОЗМІРУ ПС, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ YII

2.1 Аналіз емпіричних даних з метою виявлення та усунення викидів для побудови математичної моделі для прогнозування розміру ПС, розроблених з використанням фреймворку Yii

Для побудови нелінійної регресійної моделі для LOC-оцінювання необхідний репрезентативний набір реальних даних. У цьому дослідженні емпіричну основу складуть відкриті проєкти, розроблені з використанням фреймворку Yii. Джерелом даних виступає платформа GitHub – найбільший хостинг відкритого програмного коду, що дозволяє отримати доступ до великої кількості реальних проєктів різного масштабу та призначення.

Збір та обчислення метрик проводитиметься за допомогою спеціалізованого інструменту статичного аналізу PHPMetrics. Цей інструмент надає комплексну оцінку якості коду для PHP-проєктів, включаючи більшу частину метрик CK, а також базові метрики обсягу, такі як кількість рядків коду (LOC). PHPMetrics був обраний через його здатність аналізувати структуру коду на основі абстрактного синтаксичного дерева (AST), що забезпечує точний розрахунок логічних метрик (наприклад, WMC), а не просто фізичний підрахунок рядків. Процес збору даних включатиме:

1) ідентифікацію та клонування репозиторіїв, що використовують Yii;
2) запуск PHPMetrics для кожного проєкту для отримання CSV- або JSON-звіту;
3) агрегацію результатів у єдиний набір даних для подальшого аналізу.

За результатами цього етапу було знайдено 55 проєктів веб-систем, розроблених з використанням фреймворку Yii, перелік яких наведено в таблиці 2.1.

Таблиця 2.1 – Зібрані метрики

№	Проєкт	Y	X_1	X_2	X_3	SMD
1	2	3	4	5	6	7
1	https://github.com/yiisoft/yii2-jui.git	1,44	19	2,05	2,67	3,99
2	https://github.com/sirinibin/Yii2-RESTful-API-with-OAuth2.git	1,658	31	3,65	2,2	0,74
3	https://github.com/frostealth/yii2-aws-s3.git	1,665	23	4,91	1,22	3,22
4	https://github.com/akiraz2/yii2-blog.git	1,68	25	4,96	1,82	0,40
5	https://github.com/myloveGy/examination.git	1,873	28	3,64	2,43	1,46
6	https://github.com/arogachev/yii2-excel.git	1,928	28	3,61	2,5	1,79
7	https://github.com/zhuravljov/yii2-queue-monitor.git	1,948	23	6,52	2,08	0,81
8	https://github.com/loveorigami/yii2-notification-wrapper.git	2,005	46	1,67	2,14	2,34
9	https://github.com/yii2tech/illuminate.git	2,012	17	6,41	1,88	0,75
10	https://github.com/mikemadisonweb/yii2-rabbitmq.git	2,051	16	6	1,64	1,09
11	https://github.com/yii2mod/yii2-rbac.git	2,095	22	4,59	2,33	0,82
12	https://github.com/thecodeholic/Yii2-Youtube-Clone.git	2,191	40	4,35	2	0,28
13	https://github.com/yiisoft/queue.git	2,302	58	3,41	1,12	4,37
14	https://github.com/yiisoft/yii2-httpclient.git	2,36	20	5,2	1,89	0,32
15	https://github.com/xuguoliangjj/yii2-admin-theme.git	2,384	38	3,61	2,15	0,67
16	https://github.com/akbarjoudi/yii2-bot-telegram.git	2,482	53	1,57	4	21,87
17	https://github.com/2amigos/qrcode-library.git	2,613	46	2,48	1,4	3,16
18	https://github.com/thecodeholic/yii2-ecommerce-website.git	2,662	53	3,68	2,27	0,86
19	https://github.com/urbanindo/yii2-queue.git	2,76	24	5,71	2,11	0,44
20	https://github.com/yiisoft/yii2-redis.git	3,344	10	10,4	1,89	6,13
21	https://github.com/yiisoft/yii2-gii.git	3,377	15	12,07	2,11	10,46
22	https://github.com/mdmsoft/yii2-admin	3,525	39	4,95	2	0,21
23	https://github.com/yiisoft/yii2-bootstrap4.git	3,699	26	4	2,29	0,84
24	https://github.com/raoul2000/yii2-workflow.git	3,724	24	5,25	1,44	1,74
25	https://github.com/EleTeam/UniShopX.git	4,045	72	3,44	0,7	8,69
26	https://github.com/dektrium/yii2-user.git	4,287	62	3,94	1,82	0,48
27	https://github.com/callmez/yii2-wechat.git	4,49	64	3,92	1,96	0,35
28	https://github.com/windhoney/yii2-rest-rbac.git	4,497	40	6,25	2	0,55
29	https://github.com/yiisoft/yii2-elasticsearch.git	4,531	14	12,86	2,11	12,73
30	https://github.com/yongshengli/yiicms.git	4,67	74	4,12	2,13	0,38
31	https://github.com/xiongchuan86/openadm-yii2.git	4,678	35	6,2	1,89	0,56
32	https://github.com/yiisoft/yii2-debug.git	4,776	46	4,98	2	0,20
33	https://github.com/kartik-v/yii2-grid.git	4,779	33	3,67	1,75	0,86
34	https://github.com/bedezign/yii2-audit.git	4,997	95	3,55	2	0,46

Кінець таблиці 2.1

1	2	3	4	5	6	7
35	https://github.com/yiisoft/yii2-authclient.git	5,117	35	6,57	2,75	3,71
36	https://github.com/MacGyer/yii2-materializecss.git	5,295	39	3,85	2,17	0,61
37	https://github.com/iiYii/getyii.git	6,57	132	3,76	1,96	0,32
38	https://github.com/noumo/easyii.git	6,695	146	4,05	2,05	0,28
39	https://github.com/2amigos/yii2-usuario.git	7,857	136	3,86	1,45	1,49
40	https://github.com/yiisoft/yii2-mongodb.git	8,561	36	11,56	2,21	9,07
41	https://github.com/weison-tech/yii2-cms.git	10,635	153	4,11	2	0,17
42	https://github.com/liufee/cms.git	14,616	164	4,46	1,85	0,08
43	https://github.com/luyadev/luya	17,345	160	3,41	1,72	0,75
44	https://github.com/hassiumsoft/hasscms-app.git	17,408	325	3,37	1,73	0,62
45	https://github.com/jianyan74/TinyShop.git	25,995	425	3,17	1,55	1,08
46	https://github.com/funson86/funboot.git	31,352	510	3,49	1,64	0,95
47	https://github.com/ushainformatique/whatacart.git	36,493	676	3,36	1,62	2,08
48	https://github.com/opensourcewebsite-org/opensourcewebsite-org.git	39,152	795	3,44	1,5	3,49
49	https://github.com/qmpaas/leadshop.git	43,305	417	4,58	1,42	1,11
50	https://github.com/skeeks-cms/cms.git	51,837	726	3,88	1,82	1,75
51	https://github.com/fetus-hina/stat.ink.git	85,457	2141	3,74	1,22	39,99
52	https://github.com/bearlord/kantphp2.git	93,808	367	8,71	1,31	12,16
53	https://github.com/kaidianxing/kaidianxing-shop.git	115,288	1044	3,47	1,18	7,76
54	https://github.com/humhub/humhub.git	150,188	1428	4,26	1,06	12,79
55	https://github.com/craftcms/cms.git	181,82	1268	6,40	1,81	25,70

Для побудови нелінійної регресійної моделі було обрано три ключові метрики: кількість класів (Classes X_1), середню кількість методів на клас (Avg num of methods X_2) та глибину дерева успадкування (DIT X_3). Цей вибір обґрунтований кількома факторами.

По-перше, ці метрики широко використовуються в аналогічних дослідженнях для прогнозування розміру ПЗ [7 - 10]. Наприклад, кількість класів є прямим індикатором обсягу системи, середня кількість методів характеризує внутрішню складність компонентів, а DIT відображає архітектурну складність, пов'язану з ієрархією успадкування, що є ключовою особливістю об'єктно-орієнтованих фреймворків, таких як Yii.

По-друге, важливим статистичним критерієм для багатовимірної регресії є відсутність мультиколінеарності – високої кореляції між незалежними змінними (метриками). Наявність мультиколінеарності робить модель нестійкою, ускладнює інтерпретацію впливу окремих факторів і знижує її прогностичну здатність [25]. Для зібраних даних отримано значення VIFs 1,223, 1,022 та 1,211 відповідно. Проведений попередній аналіз кореляційної матриці та розрахунок статистики VIF (Variance Inflation Factor) підтверджують, що обраний набір метрик демонструє прийнятний рівень взаємної незалежності (всі значення менше 5), що робить їх придатними для спільного використання в моделі без ризику дестабілізації оцінок.

Вибір нелінійної регресійної моделі зумовлений фундаментальною неможливістю застосування лінійних регресійних моделей для даних дослідження, оскільки вони суттєво порушують її ключові статистичні припущення. Лінійна модель вимагає, щоб залишки (а в ідеалі – і самі дані) були розподілені нормально. Однак зв’язок між архітектурними метриками (наприклад, середньою кількістю методів) та обсягом коду (LOC) у реальних проєктах на Yii є суттєво нелінійним: зростання складності призводить не до пропорційного, а до прискореного збільшення обсягу коду через додавання внутрішніх зв’язків та шаблонного коду фреймворку.

На основі обраних метрик розраховуємо квадрат відстані Махаланобіса [26] для кожного спостереження.

В цій формулі значення під знаком суми – це квадрат відстані Махаланобіса, і-ту точку якого можна розрахувати як

$$SMD_i = (X_i - \overline{X})^T S_N^{-1} (X_i - \overline{X}) , \qquad (2.1)$$

де X_i – i- та точка вибірки,

$\overline{X}$ – вектор середніх значень,

S_N – коваріаційна матриця,

N – розмір вибірки, вектор чотирьох компонентний.

Коваріаційну матрицю можна отримати як

$$S_N = \frac{1}{N}\sum_{i=1}^{N}(X_i - \overline{X})\,(X_i - \overline{X})^T. \quad (2.2)$$

За результатами розрахунку квадрата відстані Махаланобіса з таблиці 2.1 бачимо, що для точок 16, 51, та 55 отримане значення більше за критичне значення квантіля критерія хі-квадрат 14,86 для рівня значущості 0,005.

Оскільки припущення про нормальний розподіл емпіричних даних з метрик не справдилось, це остаточно виключає можливість використання лінійної регресії та обґрунтовує перехід до нелінійних методів.

Для подальшої нормалізації даних обрано багатовимірне перетворення Бокса-Кокса (multivariate Box-Cox transformation). Ключовою причиною такого вибору є необхідність врахування кореляційної структури між обраними метриками (Classes, Avg num of methods, DIT). На відміну від одномірних перетворень, застосування багатовимірних перетворень знаходить оптимальні параметри трансформації одночасно для всіх змінних, зберігаючи їхні взаємозв'язки, що досить важливо для подальшого аналізу. Даний підхід має підтверджену ефективність у дослідженнях [4, 9, 10] та є менш параметрично складним, ніж альтернативне перетворення Джонсона, що робить його більш ефективним з точки зору трудомісткості застосування при збереженні достатньої гнучкості для досягнення розподілу, близького до нормального.

Багатовимірне перетворення Бокса-Кокса можна представити як

$$Z_j = \begin{cases} \frac{\left(X_j^{\lambda_j} - 1\right)}{\lambda_j}, \text{якщо } \lambda_j \neq 0; \\ \ln X_j, \text{якщо } \lambda_j = 0. \end{cases}, \quad (2.3)$$

де Z_j – нормалізована точка даних,

λ_j – параметр нормалізуючого перетворення Бокса-Кокса.

Для вилучення викидів у багатовимірних емпіричних даних слід застосувати до них обране нормалізуюче перетворення разом з формулами (2.1) та (2.2) для

визначення квадрату відстані Махаланобіса для нормалізованих даних. У випадку, коли хоча б одна точка перевищує критичне значення (14,86) для рівня значущості 0,005, слід вилучити точку з найбільшим значення SMD_i. Повторювати до тих пір, поки усі такі точки даних не будуть усунуті [27, 28].

Серед емпіричних даних було знайдено лише один викид – проєкт №16.

2.2 Математична модель для прогнозування розміру ПС, розроблених на Yii

Алгоритм побудови нелінійної регресійної моделі на основі нормалізуючих перетворень виконує такі дії [29, 30]:

1) Дані перетворюються за обраним нормалізуючим перетворенням.

2) Видаляються викиди, після чого будується лінійна регресія.

3) Залишки регресії перевіряються на нормальність критерієм Пірсона.

4) Якщо умова не виконується, виключається рядок з найбільшим за модулем залишком, і кроки 1-3 повторюються.

5) Після досягнення нормальності залишків виконується зворотне перетворення для отримання нелінійної моделі. Будується інтервал прогнозування; дані за його межами виключаються.

Нелінійна регресійна модель з застосування нормалізуючого перетворення Бокса-Кокса має вигляд

$$Y = \left[\hat{\lambda}_Y\left(\hat{Z}_Y + \varepsilon\right) + 1\right]^{1/\hat{\lambda}_Y}, \tag{2.4}$$

де $\hat{Z}_Y$ – результат розрахунку линійного рівняння регресії $\hat{Z}_Y = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \hat{b}_3 Z_3$ для нормалізованих даних за використанням (2.4),

ε – нормально розподілена випадкова величина.

Таблиця 2.2 містить інформацію про побудову нелінійної регресійної моделі на кожній з ітерацій

Таблиця 2.2 – Побудова нелінійної регресійної моделі

Ітерація	Кількість проєктів	SMD_{max}	Параметри перетворення	Параметри регресії	Викиди
1	55	18,927	$\lambda_Y = -0{,}1742$ $\lambda_{X_1} = -0{,}0991$ $\lambda_{X_2} = 0{,}0939$ $\lambda_{X_3} = 0{,}5551$	–	1, SMD (проєкт №16)
2	54	12,549	$\lambda_Y = -0{,}1656$ $\lambda_{X_1} = -0{,}0935$ $\lambda_{X_2} = 0{,}0061$ $\lambda_{X_3} = 1{,}6082$	$b_0 = -3{,}184360$ $b_1 = 1{,}017821$ $b_2 = 0{,}746962$ $b_3 = 0{,}019366$	1, інтервал прогнозування (проєкт № 33)
3	53	12,584	$\lambda_Y = -0{,}1671$ $\lambda_{X_1} = -0{,}0861$ $\lambda_{X_2} = 0{,}0647$ $\lambda_{X_3} = 1{,}6470$	$b_0 = -3{,}173005$ $b_1 = 0{,}996457$ $b_2 = 0{,}696808$ $b_3 = 0{,}035899$	1, ε не є $N(0, \sigma_\varepsilon^2)$ (проєкт № 36)
4	52	12,964	$\lambda_Y = -0{,}1701$ $\lambda_{X_1} = -0{,}0797$ $\lambda_{X_2} = 0{,}1217$ $\lambda_{X_3} = 1{,}6012$	$b_0 = -3{,}068298$ $b_1 = 0{,}963726$ $b_2 = 0{,}640257$ $b_3 = 0{,}026629$	1, ε не є $N(0, \sigma_\varepsilon^2)$ (проєкт № 23)
5	51	14,000	$\lambda_Y = -0{,}1708$ $\lambda_{X_1} = -0{,}0820$ $\lambda_{X_2} = 0{,}2000$ $\lambda_{X_3} = 1{,}5504$	$b_0 = -3{,}060299$ $b_1 = 0{,}979900$ $b_2 = 0{,}575199$ $b_3 = 0{,}010813$	1, ε не є $N(0, \sigma_\varepsilon^2)$ (проєкт № 43)
6	50	13,961	$\lambda_Y = -0{,}1785$ $\lambda_{X_1} = -0{,}0701$ $\lambda_{X_2} = 0{,}2324$ $\lambda_{X_3} = 1{,}5642$	$b_0 = -2{,}879428$ $b_1 = 0{,}907603$ $b_2 = 0{,}541000$ $b_3 = 0{,}014299$	1, ε не є $N(0, \sigma_\varepsilon^2)$ (проєкт № 1)
7	49	11,686	$\lambda_Y = -0{,}1421$ $\lambda_{X_1} = -0{,}0761$ $\lambda_{X_2} = 0{,}1466$ $\lambda_{X_3} = 1{,}6515$	$b_0 = -3{,}407658$ $b_1 = 1{,}024284$ $b_2 = 0{,}726713$ $b_3 = -0{,}026164$	1, ε не є $N(0, \sigma_\varepsilon^2)$ (проєкт № 13)
8	48	10,975	$\lambda_Y = -0{,}1422$ $\lambda_{X_1} = -0{,}0828$ $\lambda_{X_2} = 0{,}1811$ $\lambda_{X_3} = 1{,}6481$	$b_0 = -3{,}320265$ $b_1 = 1{,}043260$ $b_2 = 0{,}674595$ $b_3 = -0{,}061035$	–

Для комплексної оцінки якості та достовірності побудованої нелінійної регресійної моделі будуть використані стандартні метрики, такі як:

– MMRE (Mean Magnitude of Relative Error): середня абсолютна величина відносної похибки. Ця метрика оцінює середню точність прогнозів моделі. Чим менше значення MMRE, тим вища загальна точність.

– PRED(0.25) (Prediction at level 0.25): частка прогнозів, відносна похибка яких не перевищує 25%. Цей показник демонструє практичну придатність моделі, відображаючи процент прогнозів, що є достатньо точними для реального планування.

– R^2 (Коефіцієнт детермінації): вимірює, яку частку дисперсії залежної змінної (LOC) пояснює модель. Вище значення R^2 вказує на кращу здатність моделі описувати варіації у даних.

Для побудованої моделі: $R^2 = 0{,}968$, $MMRE = 0{,}131$, $PRED(0{,}25) = 0{,}833$. Отримані оцінки свідчать про високу якість моделі [31, 32].

Продемонструємо, що застосування моделей інших фреймворків мови PHP дає незадовільні результати у порівнянні з моделлю під обраний фреймворк. У таблиці 2.3 наведено порівняння оцінок R^2, $MMRE$, $PRED(0{,}25)$.

Таблиця 2.3 – Оцінки якості нелінійних регресійних моделей для різних фреймворків

Оцінки якості моделі	Модель		
	Yii	CakePHP	CodeIgniter
R^2	0,968	0,917	0,725
$MMRE$	0,131	0,269	0,375
$PRED(0{,}25)$	0,833	0,604	0,417

Результати порівняння показують, що застосування універсальних моделей або моделей, побудованих для інших фреймворків, є менш ефективним.

Застосування існуючої моделі для прогнозування розміру ПС, розроблених з використанням фреймворку Yii з роботи [24] дає низькі оцінки якості: $R^2 = 0{,}767$, $MMRE = 0{,}465$, $PRED(0{,}25) = 0{,}395$. Такий результат можна пояснити тим, що в роботі [24] для побудови моделі взято 40 проєктів (проти 55 в цій роботі), а самі проекти мають більшу ширину діапазону значень, зокрема для великих ПС, розроблених з використанням фреймворку Yii. Це в свою чергу ще раз підтверджує актуальність та достовірність побудованої моделі.

2.3 Висновки до розділу 2

У цьому розділі було проведено аналіз емпіричних даних та розроблено математичну модель для прогнозування розміру програмних систем, розроблених на фреймворку Yii. На основі 55 реальних проєктів було обґрунтовано вибір трьох ключових метрик (кількість класів, середня кількість методів, глибина дерева успадкування) та підтверджено відсутність мультиколінеарності між ними. Статистичний аналіз (квадрат відстані Махаланобіса) виявив суттєве відхилення даних від багатовимірного нормального розподілу, що остаточно виключило можливість використання лінійної регресії. Для нормалізації даних було застосовано багатовимірне перетворення Бокса-Кокса, що дозволило врахувати кореляційну структуру метрик. За допомогою ітеративного алгоритму було виключено викиди та побудовано нелінійну регресійну модель (R^2=0.968, MMRE=0.131, PRED(0.25)=0.833). Порівняння з моделями для інших PHP-фреймворків (CakePHP, CodeIgniter, існуючої моделлю для Yii) підтвердило вищу якість та практичну придатність розробленої спеціалізованої моделі для екосистеми Yii.

3 РОЗРОБКА ПРОЄКТНИХ РІШЕНЬ ДЛЯ ПРОГРАМНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ РОЗМІРУ ПС, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ YII

3.1 Постановка задачі на розробку програмної системи прогнозування розміру ПС, розроблених з використанням фреймворку Yii

На основі попередніх розділів, де була побудовано нелінійну регресійна модель для прогнозування розміру програмних систем, розроблених з використанням фреймворку Yii, виникає практична потреба в інструменті для її застосування. Для вирішення цього завдання поставлено наступну мету: розробити ПС, яке реалізує побудовану математичну модель та дозволяє автоматизувати процес прогнозування розміру нових проєктів на Yii.

Програмна система, яка отримала назву «YiiSizeEstimator», орієнтована на керівників проєктів та архітекторів. Вона покликана спростити планування ресурсів шляхом надання обґрунтованих оцінок обсягу коду вже на етапі проєктування архітектури.

Вимоги до складу виконуваних функцій

Програмна система «YiiSizeEstimator» повинна забезпечувати наступну функціональність:

– Внесення параметрів готової моделі – введення коефіцієнтів регресії (b_0, b_1, b_2, b_3) та параметрів λ для залежної та незалежних змінних (λ_Y, λ_X_1, λ_X_2, λ_X_3), отриманих в попередніх розділах.

– Введення метрик проєкту, що оцінюється від користувача (трьох ключових метрик для конкретного проєкту): кількість класів (Classes), середня кількість методів на клас (Avg num of methods) та глибина дерева успадкування (DIT).

– Автоматизований розрахунок прогнозованого розміру ПС – виконання послідовності математичних операцій, що включає: застосування багатовимірного перетворення Бокса-Кокса до вхідних метрик, розрахунок прогнозованого значення

за допомогою лінійної регресійної моделі для нормалізованих даних, виконання зворотного перетворення для отримання прогнозованого розміру (LOC) за допомогою взаємозворотнього перетворення. Програмна система також проводить розрахунок інтервальних оцінок, зокрема довірчого інтервалу та інтервалу прогнозування прогнозованого значення.

– Наочне представлення результатів – виведення користувачеві кінцевих результатів у зрозумілому форматі:

1) Очікуваного розміру проєкту (LOC).
2) Нижньої та верхньої меж довірчого інтервалу.
3) Нижньої та верхньої меж інтервалу прогнозування.

Вимоги до інформаційної та програмної сумісності

– Мова програмування: Python 3.8 або вище.

– Ключові бібліотеки: NumPy, SciPy, pandas, scikit-learn для математичних та статистичних обчислень.

– Інтерфейс – простий графічний інтерфейс на основі бібліотеки Qt для введення даних та виведення результатів.

– Операційна система – крос-платформне рішення, що працює під управлінням ОС Windows, Linux, macOS.

Вимоги до складу та параметрів технічних засобів:

Для функціонування програмної системи достатніми є мінімальні технічні характеристики:

– Процесор з архітектурою x86-64. 512 МБ оперативної пам'яті.

– 100 МБ вільного місця на носії даних для встановлення інтерпретатора Python та необхідних бібліотек.

Детальна специфікація вимог та технічне завдання на розробку програмної системи «YiiSizeEstimator» представлені у Додатку А.

3.2 Загальносистемні рішення програмної системи прогнозування розміру ПС, розроблених з використанням фреймворку Yii

3.2.1 Вибір засобів моделювання програмної системи для прогнозування розміру ПС

Для забезпечення чіткої структуризації, високої якості та зручності підтримки розроблюваної програмної системи «YiiSizeEstimator» обрано об’єктно-орієнтовану (ОО) методологію. Цей підхід дозволяє природно відобразити основні сутності предметної області у вигляді класів, інкапсулювати логіку їх поведінки та визначити чіткі інтерфейси взаємодії між компонентами системи.

У рамках ОО методології для візуалізації, специфікації та документування архітектури системи використовується стандарт UML (Unified Modeling Language) [33]. UML є універсальною та широковизнаною мовою моделювання, яка дозволяє наочно представити різні аспекти системи, від функціональних вимог користувача до фізичного розгортання компонентів.

Архітектура системи та процес її розробки будуть детально описані за допомогою наступного комплекту діаграм та моделей UML, структурованих відповідно до загальноприйнятих рекомендацій:

Діаграми, що описують загальносистемні рішення та функціональність:

1) Діаграма варіантів використання для визначення взаємодії акторів (користувача) з системою.

2) Специфікації та сценарії основних варіантів використання для детального опису логіки роботи.

Діаграми, що визначають інформаційне забезпечення:

1) Концептуальна модель даних, що відображає ключові сутності та зв'язки між ними.

2) Логічна модель даних для проєктування структури зберігання інформації.

Діаграми, що деталізують рішення з програмного забезпечення та архітектуру:

1) Модель класів, що є центральною в ОО проєктуванні та описує статичну структуру системи.

2) Моделі взаємодії (наприклад, діаграми послідовностей) для опису динаміки співпраці об'єктів.

3) Модель компонентів для відображення фізичного поділу системи на модулі.

4) Модель розгортання для опису конфігурації середовища виконання.

5) Фізична модель даних для остаточного визначення схем зберігання.

Такий комплексний підхід до моделювання за допомогою UML дозволить створити узгоджену, документовану та технологічно обґрунтовану архітектуру, що є основою для успішної реалізації програмної системи «YiiSizeEstimator».

3.2.2 Розробка діаграми варіантів використання програмної системи прогнозування розміру ПС

Діаграма варіантів використання (use case diagram) є основним інструментом UML для визначення функціональних вимог до системи з точки зору зовнішніх акторів. Вона описує, що повинна робити система, але не як саме, фокусуючись на взаємодії між користувачами (акторами) та системою у вигляді цілісних функціональних блоків – варіантів використання.

Для програмної системи «YiiSizeEstimator» основним та єдиним актором є Користувач – керівник проєкту, архітектор або розробник, який потребує отримати прогноз розміру ПС.

На основі сформованих вимог у попередньому підрозділі, визначено наступні варіанти використання, які наведено у таблиці 3.1.

Таблиця 3.1 – Визначені прецеденти

Основні актори	Найменування	Формулювання
Користувач	Ввести параметри готової моделі.	Користувач задає коефіцієнти та параметри λ попередньо побудованої математичної моделі.
Користувач	Ввести метрики проєкту.	Користувач вказує значення трьох ключових метрик (кількість класів, середня кількість методів, DIT) для конкретного проєкту, що оцінюється.
Користувач	Виконати розрахунок прогнозу.	Система, отримавши всі необхідні вхідні дані, автоматично виконує послідовність обчислень за математичною моделлю (нормалізація, лінійна регресія, зворотне перетворення).
Користувач	Отримати прогнозований розміру.	Система надає користувачу результати розрахунку у зручному вигляді: точковий прогноз розміру (LOC), межі довірчого інтервалу та межі інтервалу прогнозування.

Рисунок 3.1 містить побудовану use-case діаграму.

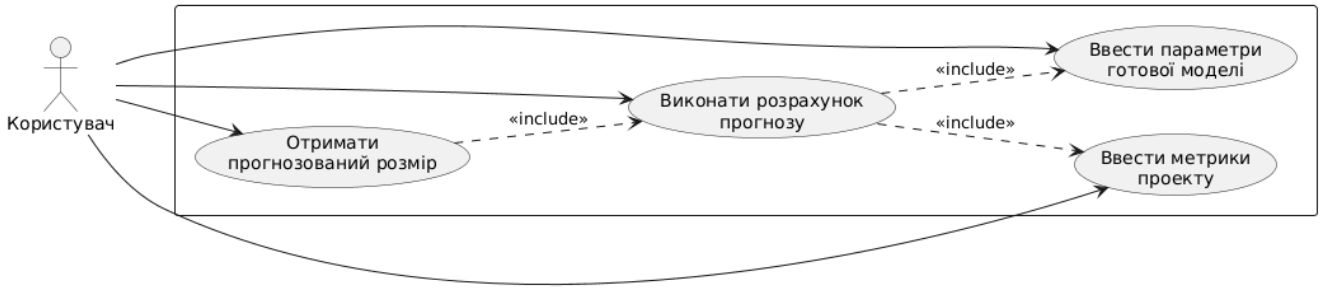

Рисунок 3.1 – Діаграма варіантів використання

Ці варіанти використання формують логічну послідовність дій користувача для отримання кінцевого результату. Їх взаємозв'язок та відношення до актора представлені на діаграмі варіантів використання, яка наочно ілюструє весь спектр функціональних можливостей системи.

3.2.3 Створення діаграм діяльності програмної системи для прогнозування розміру ПС

Діаграма діяльності (activity diagram) є одним з основних інструментів UML для моделювання бізнес-процесів та алгоритмічної логіки. Вона візуалізує потік управління від дії до дії, що особливо ефективно для опису послідовних та паралельних операцій у системі. Можна виділити такі елементи діаграми: початковий та кінцевий вузли, дії (прямокутники з закругленими кутами), рішення (ромби), перехідні стрілки, а також об'єкти даних та розгалуження для паралельних потоків. Для комплексного опису функціонування програмної системи «YiiSizeEstimator» розроблено універсальну діаграму діяльності, яка інтегрує інтерфейс користувача, логіку валідації та математичне ядро в єдиний процес.

Діаграму діяльності показано на рисунку 3.2.

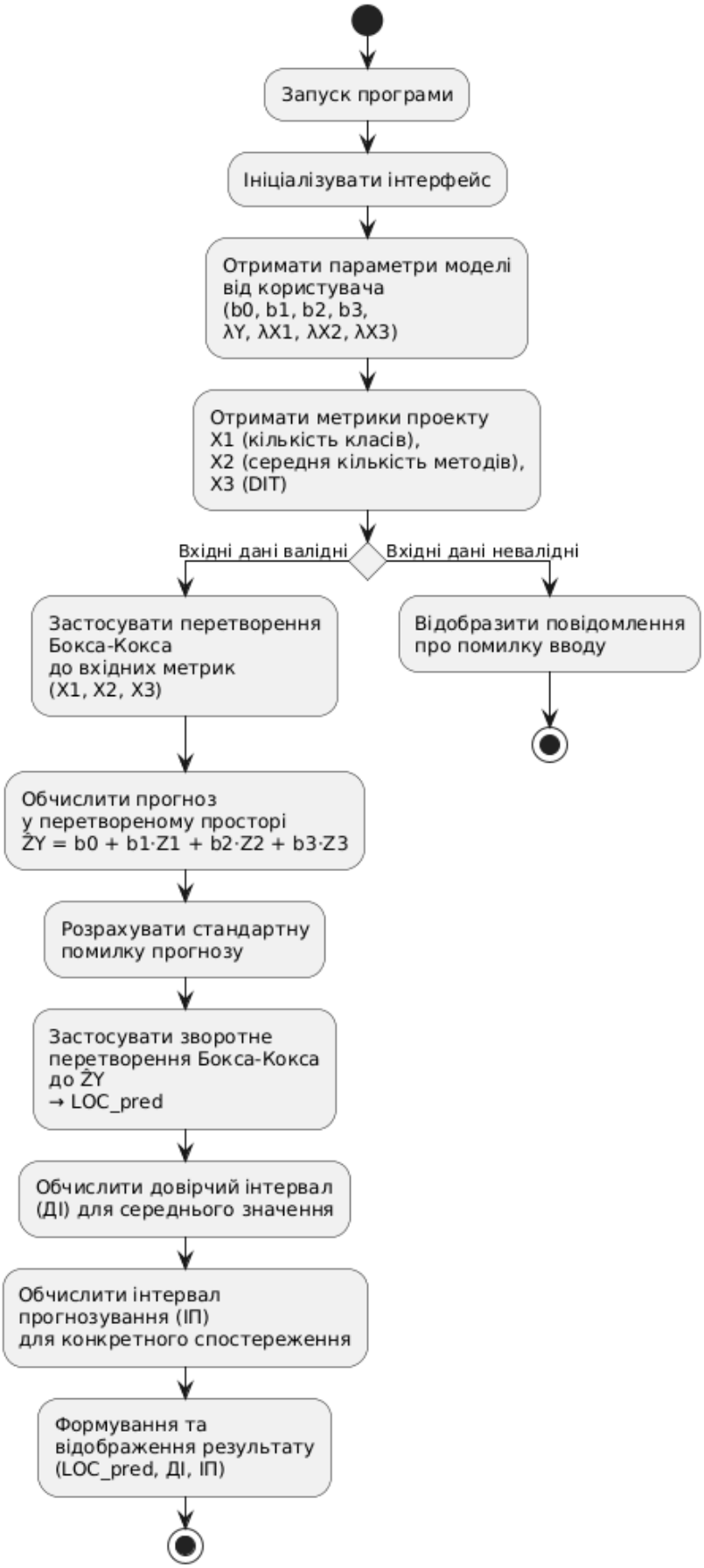

Рисунок 3.2 – Діаграма діяльності

Ця діаграма повністю описує логіку роботи програмної системи, демонструючи послідовність кроків від отримання вхідних даних до формування кінцевого прогнозу з оцінкою точності.

3.3 Рішення з інформаційного забезпечення програмної системи для прогнозування розміру ПС

Інформаційне забезпечення програмної системи «YiiSizeEstimator» визначає структуру та семантику даних, що обробляються, і є основою для реалізації її логіки. Воно формалізується в контексті інформаційної моделі, яка описує ключові сутності, їх атрибути та взаємозв'язки в контексті процесу прогнозування.

Вхідні дані системи поділяються на дві логічні групи:

1) Параметри математичної моделі: це константні значення, отримані в результаті попереднього навчання нелінійної регресійної моделі. До них належать:

– Коефіцієнти регресійного рівняння b_0, b_1, b_2, b_3.

– Параметри λ багатовимірного перетворення Бокса-Кокса для кожної змінної.

– Статистичні характеристики навчальної вибірки: вектор середніх значень трансформованих змінних та обернена коваріаційна матриця, необхідні для розрахунку інтервальних оцінок.

2) Оперативні дані користувача: змінні значення, що вводяться для конкретного проєкту, що оцінюється:

– Кількість класів (X_1) – ціле додатне число.

– Середня кількість методів на клас (X_2) – дійсне додатне число.

– Глибина дерева успадкування (X_3) – дійсне невідёємне число.

– Рівень довіри – дійсне число в інтервалі (0, 1), наприклад, 0.95.

Вихідні дані системи є результатом застосування математичної моделі до вхідних даних і включають:

– Точковий прогноз розміру ПС – очікувана кількість рядків коду (LOC), дійсне число.

– Довірчий інтервал для прогнозу – нижня та верхня межа (LOC), що характеризують точність оцінки середнього значення.

– Інтервал прогнозування – нижня та верхня межа (LOC), що визначають діапазон можливих значень для конкретної оцінюваної системи.

3.4 Рішення з програмного забезпечення програмної системи для прогнозування розміру ПС

В даному підрозділ було виконано реалізацію, тестування та випробування програмної системи для прогнозування розміру ПС, розроблених з використанням фреймворку Yii. Код розробленої системи наведено у додатку Г.

3.4.1 Засоби розробки програмної системи прогнозування розміру ПС

Для реалізації програмної системи «YiiSizeEstimator» обрано мову програмування Python та екосистему її науково-обчислювальних бібліотек. Цей вибір є стратегічним та обґрунтованим специфікою розроблюваної системи, основним завданням якої є точне виконання послідовності складних математичних перетворень та статистичних розрахунків. Python як інструмент забезпечує оптимальний баланс між продуктивністю розробки, надійністю обчислень та зручністю майбутнього використання.

Першочерговою причиною є панівне становище Python у сфері Data Science та наукових досліджень. Це робить його природним вибором для реалізації прогнозних моделей, оскільки велика частина сучасних алгоритмів машинного навчання, статистичного аналізу та обробки даних реалізована саме на цій мові. Відповідно, існує потужна та зріла екосистема бібліотек, спеціально оптимізованих для подібних завдань. Так, для ефективної роботи з матричними операціями та багатовимірними масивами, які є основою регресійних обчислень, використовуватиметься бібліотека NumPy. Бібліотека SciPy буде залучена для

розрахунку статистик розподілу, адже бібліотека що містить перевірені, академічно коректні реалізації, мінімізуючи ризик алгоритмічних помилок «власної розробки».

Крім того, синтаксис Python відрізняється лаконічністю та читаністю, що суттєво прискорює процес написання, налагодження та супроводу програмного коду, особливо коли він містить складні математичні формули. Це також спрощує можливість подальшого розширення функціоналу системи, наприклад, додавання модуля для аналізу наборів даних з бібліотеки Pandas чи впровадження додаткових метрик якості з використанням scikit-learn.

Важливим практичним аспектом є крос-платформність Python. Готова програмна система може бути легко запущене в будь-якій сучасній операційній системі без необхідності адаптації вихідного коду. Для створення зручного графічного інтерфейсу, який дозволить користувачеві вводити параметри моделі та метрики проєкту, можна використати бібліотку PythonQt, що робить систему доступною для користувачів без технічної підготовки.

Таким чином, вибір Python як основної платформи розробки є цілком виправданим. Він забезпечує не лише технічну можливість точної реалізації математичної моделі, але й створює підґрунття для створення зручного, надійного та легко підтримуваного інструменту, що повністю відповідає основним цілям проєкту.

3.4.2 Модель класів для програмної системи прогнозування розміру ПС

Модель класів (class diagram) UML є фундаментальною статичною структурною діаграмою, що визначає каркас програмної системи. Вона формалізує ключові абстракції предметної області у вигляді класів, визначає дані, які вони зберігають (атрибути), операції, які вони виконують (методи), та найважливіше – взаємозв'язки між ними. Для системи «YiiSizeEstimator», основним призначенням якої є виконання детермінованого математичного процесу, модель класів слугує формальним описом архітектури, яка забезпечує модульність, зрозумілість та легкість супроводу.

Модель класів для «YiiSizeEstimator» реалізує варіацію патерну «Сервісний шар» (Service Layer), де бізнес-логіка (математичні обчислення) інкапсульована в окремі сервісні класи, а дані представлені сутнісними класами-моделями. Це дозволяє чітко відокремити логіку перетворення даних від власне даних та інтерфейсу користувача. Система організована навколо логічного потоку даних: завантаження моделі → введення метрик → розрахунок → результат.

Модель класів показано на рисунку 3.3.

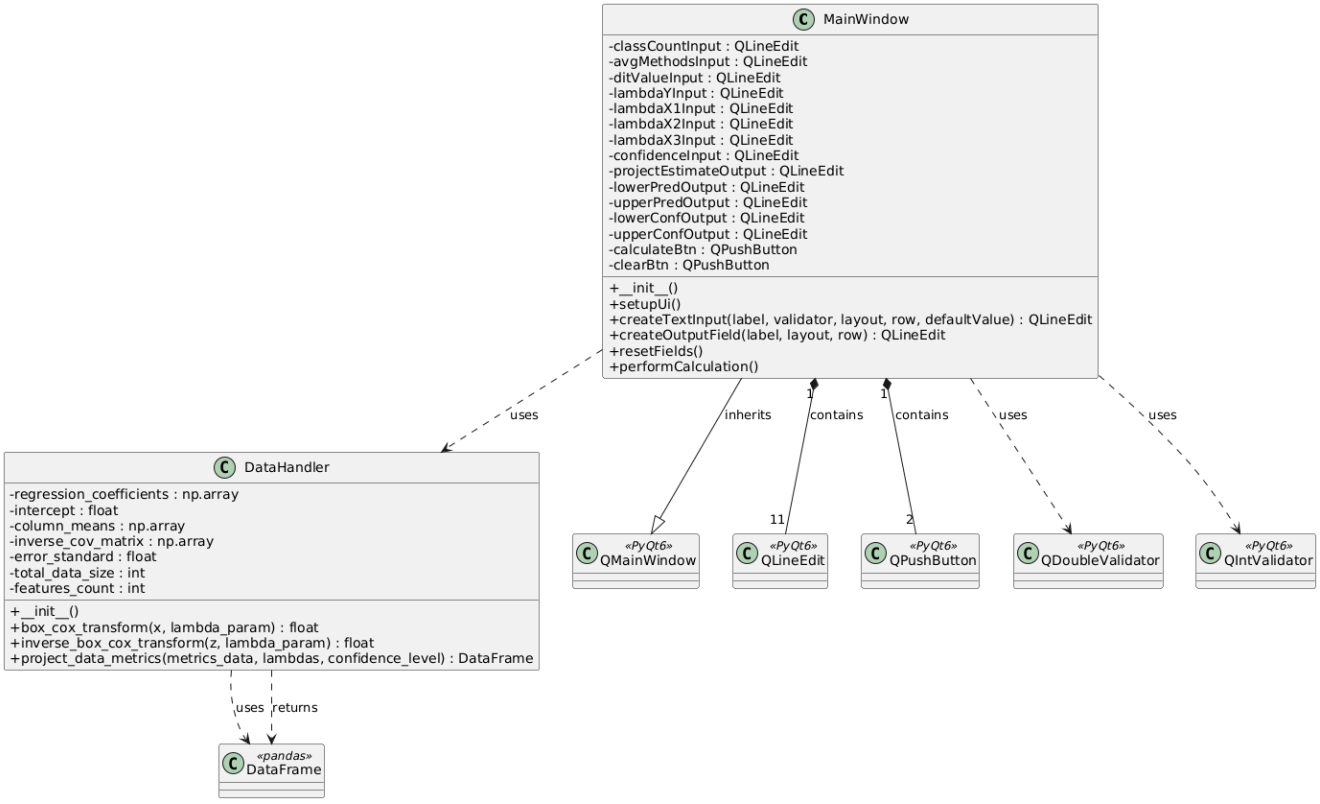

Рисунок 3.3 – Модель класів

Така деталізована модель класів не тільки слугує керівництвом для реалізації, але й формально документує архітектурні рішення, що забезпечують точність, надійність та зручність розширення програмної системи «YiiSizeEstimator».

Наведемо короткий огляд основних елементів діаграми класів.

MainWindow (Рівень користувацького інтерфейсу) – це графічний інтерфейс, з яким взаємодіє користувач. Він містить:

– Поля введення (8 віджетів QLineEdit) – для введення програмних метрик (класи, методи, DIT) та параметрів лямбда Box-Cox

– Поля виведення (5 віджетів QLineEdit) – відображають розраховану оцінку LOC та довірчі/прогнозні інтервали

– Кнопки (2 віджети QPushButton) – «Розрахувати» запускає обчислення, «Очистити» скидає всі поля

– Методи – відповідають за створення UI, взаємодію з користувачем, валідацію введення та координацію з Evaluator

Evaluator (Рівень бізнес-логіки) – це обчислювальний рушій, який виконує всі математичні операції. Містить:

– Параметри моделі (зберігаються як атрибути) – попередньо розраховані коефіцієнти регресії, середні значення, матриця коваріації, стандартна похибка – це «мозок» моделі.

– Методи трансформації – box_cox_transform() та inverse_box_cox_transform() перетворюють дані між оригінальним та нормалізованим простором

– Основний метод розрахунку – project_data_metrics() приймає вхідні метрики, застосовує трансформацію Box-Cox, розраховує прогноз LOC зі статистичними інтервалами.

Зовнішні залежності

– Класи PyQt6 (QMainWindow, QLineEdit, QPushButton, валідатори) – забезпечують фреймворк GUI

– DataFrame pandas – використовується для організації вхідних/вихідних даних у табличному форматі

Потік виконання виглядає наступним чином: користувач вводить дані → MainWindow валідує → Evaluator розраховує → MainWindow відображає результати

Це розділення дозволяє відокремити код інтерфейсу від логіки розрахунків, що робить системи простішою для супроводу та тестування.

3.4.3 Моделі взаємодії для програмної системи прогнозування розміру ПС

Діаграми взаємодії (interaction diagrams) у UML слугують для моделювання динамічної поведінки системи, показуючи, як об'єкти спілкуються між собою для виконання конкретної задачі у часі. Найбільш поширеним типом є діаграма послідовності (sequence diagram), яка візуалізує потік повідомлень (викликів методів) між об'єктами в хронологічному порядку. Для системи «YiiSizeEstimator» діаграма послідовності критично важлива, оскільки демонструє, як компоненти, визначені в статичній моделі класів, кооперуються для реалізації основного сценарію використання — отримання прогнозу.

Основні елементи діаграми послідовності та їх роль:

– Об'єкти (Participants) – прямокутники вгорі діаграми, що представляють екземпляри класів.

– Життєва лінія (lifeline) кожного об'єкта показана вертикальною пунктирною лінією.

– Повідомлення (Messages) – стрілки між життєвими лініями, що позначають виклики методів або передачу даних. Синхронні повідомлення (звичайні виклики) зображуються суцільною стрілкою, а відповідь – пунктирною.

– Прямокутники активації (Activation Bars) – тонкі прямокутники на життєвих лініях, що показують період, коли об'єкт виконує певну операцію.

– Умови та цикли (Frames) – прямокутні області з міткою (наприклад, opt для опціональної поведінки, loop для циклу, alt для альтернатив), що групують частину діаграми для відображення логіки.

Рисунок 3.4 зображає діаграму послідовності основного сценарію роботи «YiiSizeEstimator».

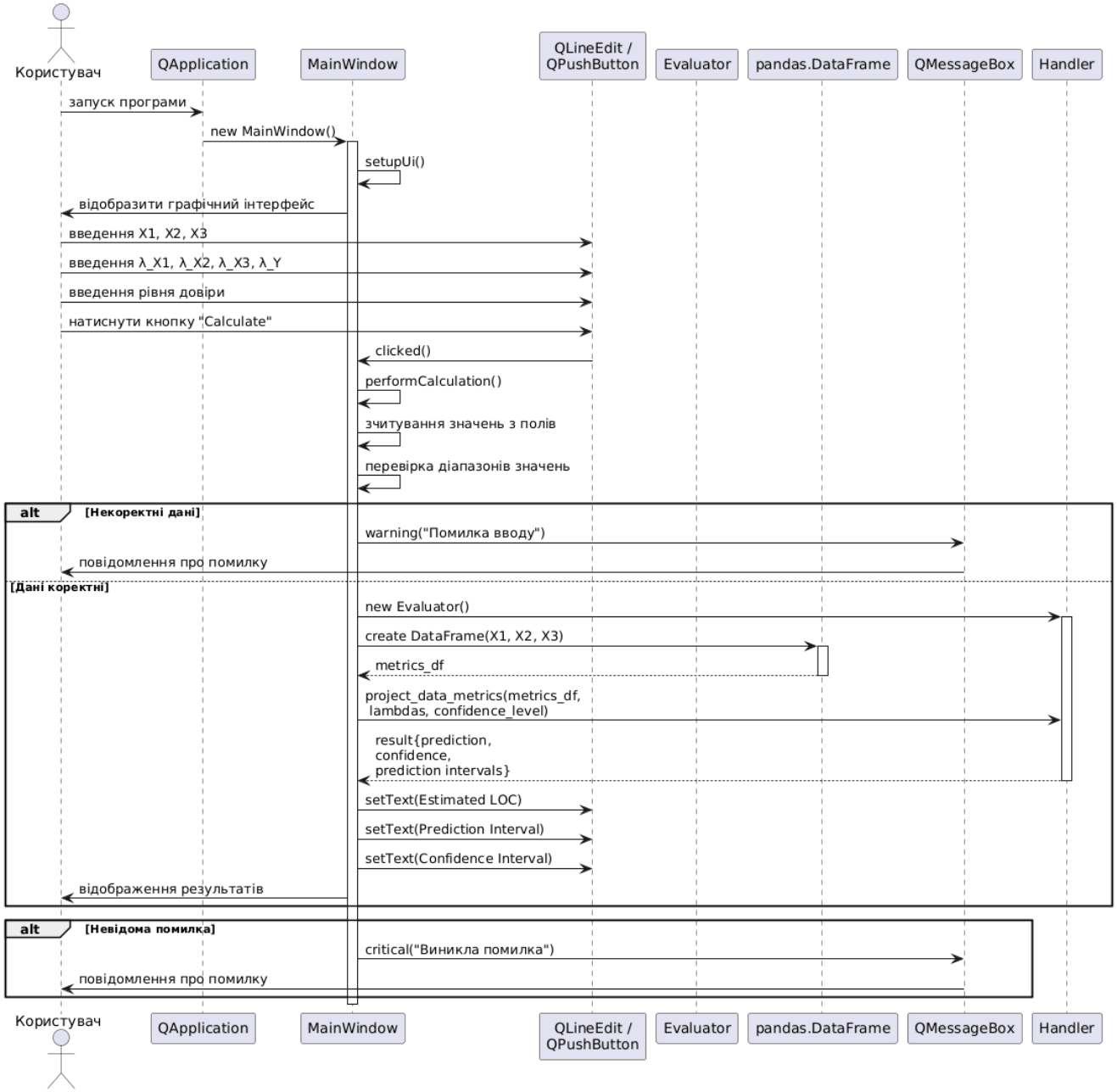

Рисунок 3.4 – Діаграма послідовності системи

Отримана модель взаємодії є корисним інструментом для перевірки коректності статичної моделі класів та для планування деталей реалізації методів у кожному класі.

3.4.4 Тестування програмної системи прогнозування розміру ПС

Тестування є критичним етапом для забезпечення якості та надійності програмної системи «YiiSizeEstimator». Оскільки система реалізує детермінований математичний алгоритм, її ядро ідеально піддається модульному (юніт) тестуванню. Для систематизації процесу тестування вибрано техніку проєктування тестів еквівалентного розбиття (equivalence partitioning).

Програмна система працює з чітко визначеними типами вхідних даних: числовими метриками проєкту (кількість класів, середня кількість методів, DIT) та параметрами математичної моделі (коефіцієнти, λ). Хоча теоретично ці числа можуть набувати будь-яких значень, на практиці логіка їх обробки має дискретні стани. Наприклад, функція валідації повинна відхиляти від'ємні значення метрик, але приймати додатні; алгоритм перетворення Бокса-Кокса використовує різні формули для випадків $\lambda=0$ та $\lambda\neq0$ і вимагає строго додатних аргументів.

Техніка еквівалентного розбиття дозволяє ефективно керувати складністю тестування в таких умовах. Замість непрактичних спроб перевірити всі можливі числові комбінації, вона пропонує згрупувати їх у класи еквівалентності – множини значень, для яких системи повинна демонструвати однакову поведінку. Це робить процес тестування систематичним і економічним: достатньо перевірити лише по кілька типових представників кожного класу, щоб з високою впевненістю зробити висновок про коректність роботи програмної системи для всієї множини схожих вхідних даних.

Таким чином, дана техніка оптимально підходить для тестування як основних розрахункових модулів системи, так і функцій валідації вхідних даних. Вона дозволяє комплексно охопити всі критичні сценарії – від нормальної роботи з коректними даними до коректної обробки помилкових або граничних значень, що є необхідною умовою створення надійної системи.

На основі вимог та архітектури системи «YiiSizeEstimator» виявлено класи еквівалентності для основних вхідних даних. Ці класи представлені в таблицях 3.2 та 3.3, а результати тестування – в таблицях 3.4, 3.5, 3.6 та 3.7.

Таблиця 3.2 показує виявлені класи еквівалентності.

Таблиця 3.2 – Виявлені класи еквівалентності для метрик проєкту

Вхідні дані	Правильні класи	Неправильні класи
Кількість класів	Натуральне число, більше 0 (1)	Дійсне неціле число (2). Від'ємне число (3). Не число (4). Пусте значення (5).
Середня кількість методів, Глибина дерева успадкування	Дійсне додатне число, більше 0 (6)	Від'ємне число (7). Не число (8). Пусте значення (9).
Параметр λ (лямбда)	Будь-яке дійсне число (10)	Не число (11). Пусте значення (12).

Результати тестування наведено в таблицях 3.3 та 3.4.

Таблиця 3.3 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Очікувана поведінка системи	Результат тестування
1	1	50	Успішна валідація, подальше виконання	Пройдено
2	6	2,25	Успішна валідація, подальше виконання	Пройдено
3	10	0,5	Успішне застосування перетворення Бокса-Кокса	Пройдено
4	10	0,0	Успішне застосування логарифмічного перетворення (окремий випадок формули)	Пройдено

Таблиця 3.4 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Очікувана поведінка системи	Результат тестування
1	2	22,2	Помилка: кількість класів має бути цілим числом	Пройдено
2	3	-10	Помилка: від'ємна кількість класів	Пройдено
3	4	«багато»	Помилка: нечислові дані	Пройдено
4	5	-	Помилка: вхідні дані відсутні	Пройдено
5	7	-1,15	Помилка: від'ємна значення	Пройдено
6	8	«багато»	Помилка: нечислові дані	Пройдено
7	9	-	Помилка: вхідні дані відсутні	Пройдено
8	11	«багато»	Помилка: нечислові дані	Пройдено
9	12	-	Помилка: вхідні дані відсутні	Пройдено

Використання техніки еквівалентного розбиття дозволяє створити компактний, але вичерпний набір модульних тестів, які перевіряють як нормальну роботу системи, так і її поведінку при помилкових або граничних вхідних даних, що є запорукою її стійкості в реальному використанні.

3.4.5 Випробування програмної системи прогнозування розміру ПС

Відповідно до затвердженої методики випробувань, наведеної у Додатку Б, було проведено комплексне випробування програмної системи «YiiSizeEstimator». У ході тестування перевірялися основні функціональні можливості системи, що реалізують процес прогнозування:

– Введення вхідних даних: коректність обробки метрик проєкту (кількість класів, середня кількість методів, глибина успадкування) та параметрів математичної моделі (λ_Y, λ_X_1, λ_X_2, λ_X_3).

– Валідація даних: спроможність системи виявляти та обробляти некоректні вхідні значення (наприклад, від'ємні числа, нечислові символи).

– Нормалізація даних: коректне застосування багатовимірного перетворення Бокса-Кокса до введених метрик згідно з заданими параметрами λ.

– Розрахункове ядро: виконання послідовності математичних операцій (лінійна регресія в перетвореному просторі та зворотне перетворення) для отримання точкового прогнозу розміру (LOC).

– Формування результатів: коректний розрахунок та відображення довірчого інтервалу та інтервалу прогнозування для заданого рівня довіри.

Процес випробування включав як перевірку штатного сценарію з коректними даними, так і ряд граничних тестів згідно з класами еквівалентності, визначеними раніше. Результати успішного виконання основного функціоналу представлено на рисунках 3.5 та 3.6.

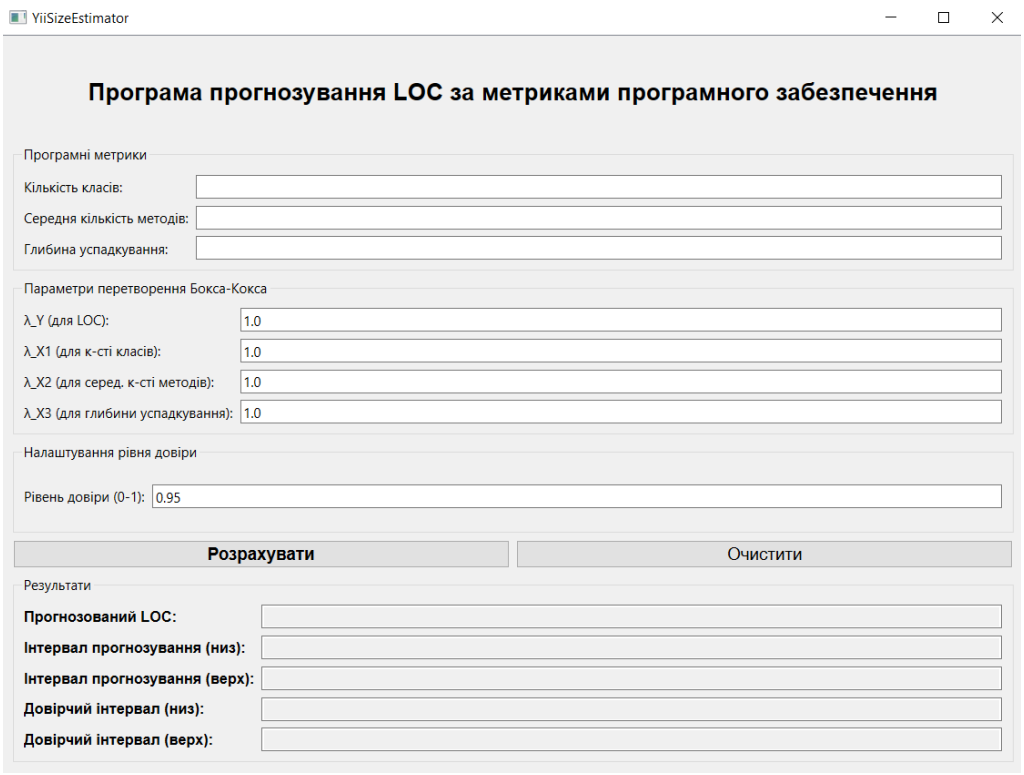

Рисунок 3.5 – Головне вікно «YiiSizeEstimator» перед початком розрахунків (введення вхідних параметрів)

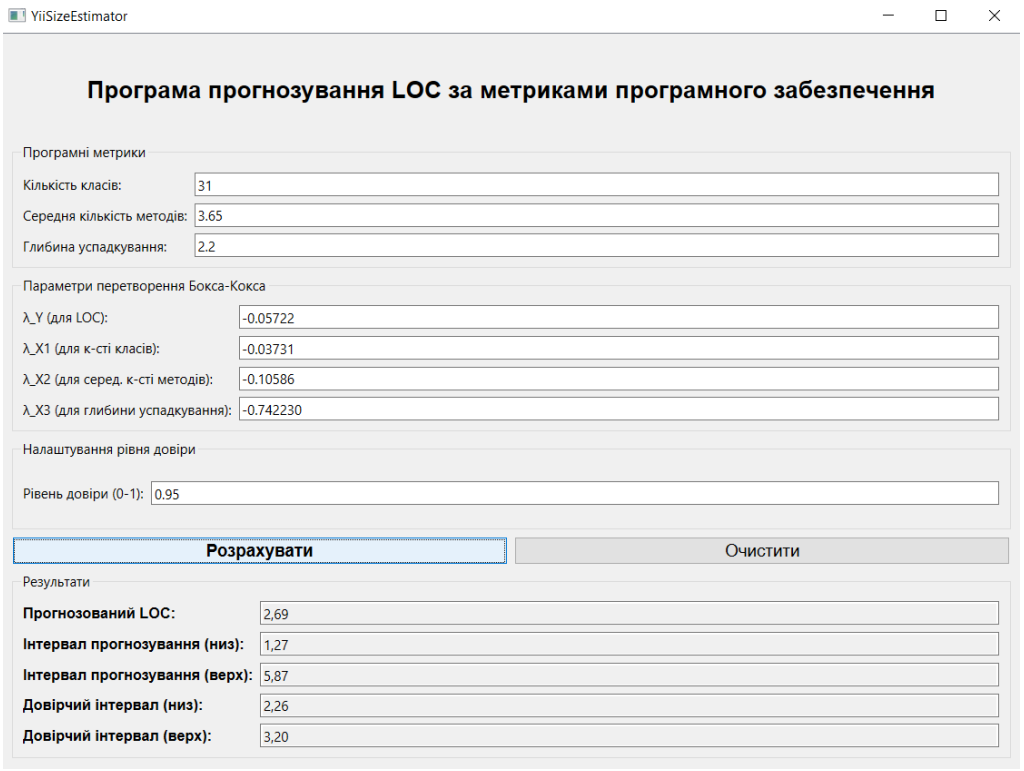

Рисунок 3.6 – Вікно «YiiSizeEstimator» після виконання прогнозування з відображенням результатів

Коректна робота системи на всіх етапах випробувань, включаючи відображення очікуваних результатів та адекватну реакцію на помилкові вхідні дані, остаточно підтверджує відповідність розробленої програмної системи всім функціональним вимогам, встановленим у технічному завданні.

3.5 Висновки до розділу 3

У цьому розділі було здійснено розробку програмної системи «YiiSizeEstimator», спрямованої на прогнозування розміру проєктів, розроблених на фреймворку Yii. На основі сформованих вимог було прийнято архітектурні рішення з використанням об’єктно-орієнтованого підходу та мови моделювання UML, що дозволило детально описати функціональність, структуру та взаємодію компонентів системи за допомогою діаграм варіантів використання, діяльності, класів та послідовностей. Вибір Python та бібліотек NumPy/SciPy забезпечив ефективну реалізацію математичного ядра системи. Розроблений комплекс модульних тестів на основі техніки еквівалентного розбиття та проведені інтеграційні випробування підтвердили високу якість, надійність та відповідність системи всім постановленим функціональним вимогам. Отримана система є готовим до використання інструментом, що автоматизує процес достовірного прогнозування розміру ПС, розроблених з використанням фреймворку Yii.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНИХ СИСТЕМ ДЛЯ ПРОГНОЗУВАННЯ РОЗМІРУ ПС, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ YII

Вступ

Інвестиції в розробку будь-якої програмної системи, включаючи спеціалізований інструмент оцінювання, повинні бути економічно обґрунтованими. Розділ, присвячений розрахунку економічної ефективності, є обов'язковою та критично важливою частиною кваліфікаційної роботи, оскільки переводить технічні результати дослідження в практичну площину бізнес-цінності. Його мета – продемонструвати, що витрати на створення програмної системи «YiiSizeEstimator» не лише окупляться, але й принесуть відчутний економічний ефект для потенційних замовників або команди розробників, що її використовуватимуть.

Розробка достовірного інструменту прогнозування розміру ПС безпосередньо впливає на два ключові драйвери вартості будь-якого ІТ-проєкту: управління ризиками та оптимізацію використання ресурсів. Неточні оцінки на ранніх етапах призводять до каскадних проблем: перевищення бюджету через неправильне планування команди, зриві дедлайнів через нереалістичні терміни, погіршення якості коду в умовах поспіху та, як наслідок, збільшення вартості підтримки. Фінансові втрати від цих проблем часто на порядки перевищують вартість розробки інструменту, що їх запобігає.

4.1 Аналіз та прорахунок витрат при створенні й експлуатації програмної системи

Вартість створення програмної системи формується з низки статей витрат. До них належать витрати на оплату праці інженера-розробника, амортизаційні відрахування та експлуатаційні витрати комп'ютерної техніки, використаної в

процесі створення, вартість ліцензійного програмного забезпечення, а також витрати на канцелярські та інші допоміжні матеріали. Роботу з реалізації програмної системи виконує фахівець, розмір щомісячного посадового окладу якого становить 30 000 гривень. Розмір доплат і надбавок до основної заробітної плати становить 20%. Таким чином, сумарний розмір основної та додаткової заробітної плати інженера складає 36 000 гривень на місяць. Ціна сучасного персонального комп'ютера, необхідного для роботи, становить 28 000 гривень. Тариф на електроенергію дорівнює 4,32 гривні за один кіловат-годину. Витрати на придбання канцелярських товарів та інших необхідних матеріалів наведено у таблиці 4.1.

Таблиця 4.1 – Кошторис витрат на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Друкований папір	500
2	Картридж та тонер для принтера	500
3	Використання флеш-накопичувача	350
4	Непередбачувані витрати	5000
	Разом	6350

Повна собівартість розробки програмної системи обчислюється за такою формулою:

$$С_{пр} = (З_{зп} + З_{зс} + З_{зг} + З_{е}) * Т + З_{м},$$

де Т – час, витрачений на розробку (місяці);

$З_{зп}$ – сума основної та додаткової заробітної плати (грн.);

$З_{зс}$ – відрахування на соціальне страхування (43% від заробітної плати) (грн.);

$З_{зг}$ – накладні витрати організації (10% від заробітної плати) (грн.);

$З_{е}$ – витрати на електроенергію (грн.);

$З_{м}$ – витрати на допоміжні матеріали (грн.).

Витрати на електроенергію (Зе) при споживаній потужності 0,5 кВт, щомісячному фонді робочого часу 22*8 = 176 годин та вказаному тарифі складають:

$$З_е = 176 * 4,32 * 0,5 = 380,16 \text{ грн.} \quad (4.1)$$

Усі поточні витрати, пов'язані з розробкою програмної системи, зведено в таблицю 4.2.

Таблиця 4.2 – Калькуляція витрат на розробку програмної системи

№	Пункти витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	2
2	Основна та додаткова заробітна плата (Ззп)	Грн.	36000
3	Відрахування на соціальні витрати (Ззс)	Грн.	15480
4	Загальногосподарські витрати (Ззг)	Грн.	3600
5	Витрати на допоміжні матеріали (Зм)	Грн.	6350
6	Витрати на електроенергію *(Зе)	Грн.	380,16

Підставивши дані з таблиці до формули (4.1), розрахуємо вартість розробки системи:

$$С_{пр} = (36000 + 15480 + 3600 + 380,16) * 2 + 6350 = 117270,32 \text{ грн.}$$

Річні амортизаційні відрахування для комп'ютерного обладнання приймаються на рівні 60% його балансової вартості:

$$А_{об} = 28000 * 0,6 = 16800 \text{ грн.}$$

На рівні підприємства річні витрати на основні та допоміжні матеріали для обслуговування техніки зазвичай становлять 5% від її вартості:

$$В_м = 28000 * 0,05 = 1400 \text{ грн.}$$

Річний фонд робочого часу персонального комп'ютера у годинах можна визначити за формулою:

$$Ф_м = 264,5 * Т_з, \quad (4.2)$$

де Тз – середньомісячне навантаження на обладнання (6 годин), а 264,5 – середня кількість робочих днів за рік.

Таким чином, річний фонд робочого часу ПК складає:

$$Ф_м = 264,5 * 6 = 1587 \text{ годин}$$

Витрати на електроенергію (Зе) за рік при такому навантаженні становлять

$$З_е = 1587 * 4,32 = 6555,84$$

Сумарні експлуатаційні витрати на ПК за рік складають:

$$З_{зр} = 16800 + 1400 + 6555{,}84 = 24755{,}84$$

Отже, загальні витрати на розробку та перший рік експлуатації програмної системи дорівнюють:

$$З_{р} = 117270{,}32 + 24755{,}84 = 142026{,}16$$

4.2 Оцінка економічної ефективності розробки та впровадження програмної системи

Пряму економічну ефективність можна визначити, виходячи з того, що використання програмної системи дозволяє оптимізувати процес і, за експертною оцінкою фахівців, вивільнити трудові ресурси одного співробітника. Річний фонд заробітної плати такого співробітника становить:

$$З = 30000 * 12 * 1 = 360000 \text{ грн.}$$

Річний економічний ефект розраховується за формулою:

$$Е_{рік} = \Delta С_{п} - Е_{п} * k, \quad (4.3)$$

де $\Delta С_{п}$ – кошти, які будуть вивільнені в результаті впровадження програмної системи (360000 грн.) за вирахуванням експлуатаційних витрат (24755,84 грн.), що дорівнює 335244,16 грн.;

$Е_{п}$ – нормативний коефіцієнт ефективності капітальних вкладень (приймається рівним коефіцієнту амортизації – 0,6);

k – одноразові капітальні витрати на розробку програмного забезпечення (117270,32 грн.).

$$Е_{рік} = 335244{,}16 - 0{,}6 * 117270{,}32 = 229194{,}16 - 70362{,}19 = 264881{,}97$$

Термін окупності програмної системи розраховується наступним чином:

$$Т = \frac{k}{\Delta С_{п}} = \frac{117270{,}32}{335244{,}16} = 0{,}35 \text{ року}$$

Таким чином, інвестиції в розробку програмної системи окупляться приблизно за 4 місяці.

4.3 Висновки до розділу 4

Розрахунок економічної ефективності демонструє високу практичну цінність розробки програмної системи «YiiSizeEstimator». Незважаючи на початкові інвестиції у розмірі близько 142 тисяч гривень, впровадження цього інструменту дозволить отримати річний економічний ефект у сумі понад 264 тисяч гривень за рахунок оптимізації процесу планування та вивільнення трудових ресурсів. Критично важливим показником є короткий термін окупності, який становить лише 4 місяці, що робить проєкт високо привабливим для інвестування та підтверджує його фінансову доцільність для компаній-розробників ПС.

5 ОХОРОНА ПРАЦІ

Успішна діяльність сучасної компанії, зокрема й у сфері розробки програмних систем, неможлива без створення безпечних та здорових умов праці для кожного працівника. Це не лише вимога законодавства, а й прямий фактор, що впливає на продуктивність, якість роботи та зменшення витрат, пов'язаних з травматизмом та професійними захворюваннями. Розділ присвячено аналізу ризиків, пов'язаних з робочим місцем інженера-програміста, та розробці комплексу заходів щодо їх мінімізації. Увага зосереджена на характерних небезпечних та шкідливих факторах, властивих роботі в офісному середовищі з інтенсивним використанням комп'ютерної техніки, а також на практичних способах забезпечення ергономіки та безпеки праці.

5.1 Аналіз умов праці та потенційних ризиків у офісному середовищі ІТ-компанії

Робоче місце фахівця з розробки ПЗ, незважаючи на відсутність явних виробничих загроз (як-от рухомі механізми або токсичні речовини), є джерелом низки специфічних факторів, які при тривалому та систематичному впливі можуть завдати шкоди здоров'ю працівника. Ці фактори умовно поділяються на фізичні, психофізіологічні та організаційні.

Фізичні фактори:

– Підвищена напруженість електромагнітних полів – монітори, системні блоки, периферійні пристрої створюють низькочастотні та електростатичні поля, тривалий вплив яких може впливати на функціонування нервової та імунної систем.

– Недостатнє або неправильне освітлення – низький рівень загальної освітленості призводить до напруги зору, тоді як надмірна яскравість екрану або блиски від джерел світла створюють дискомфорт і знижують контрастність зображення, що значно підвищує навантаження на очі.

– Підвищений рівень шуму – робота систем охолодження комп'ютерів, серверного обладнання, кондиціонерів та зовнішні джерела шуму створюють постійний акустичний фон, який сприяє підвищенню стомлюваності, роздратуванню та зниженню концентрації уваги.

– Невідповідні мікрокліматичні умови – недостатня вентиляція, низька або висока вологість повітря, некомфортна температура (занадто висока або низька) в приміщенні погіршують самопочуття та знижують працездатність.

Психофізіологічні фактори:

– Напруга зорового аналізатора (зорове стомлення) – тривала фокусування на екрані монітора, частий перехід погляду між клавіатурою, документами та екраном, мікромерцання зображення призводять до синдрому комп'ютерного зору (астенопії), що проявляється сухістю, почервонінням, різзю в очах, головним болем.

– Статичне фізичне навантаження – тривале знаходження в одній, часто незручній, позі (наприклад, сидіння за столом) викликає напругу м'язів шиї, плечового поясу, спини та рук. Це провокує розвиток скелетно-м'язових розладів, таких як остеохондроз, синдром карпального каналу, хронічні болі в спині.

– Нервово-емоційне навантаження – інтенсивна розумова праця, необхідність концентрації уваги, робота в умовах дедлайнів та частої зміни завдань створюють стійкий стресовий стан, що може призвести до емоційного вигорання, неврозів, порушень сну.

Організаційні фактори:

– Нераціональна організація робочого місця – відсутність правильної ергономіки (невідповідна висота стільця та стола, неправильне розташування монітора) значно посилює негативний вплив фізичних і психофізіологічних факторів.

– Нерегламентований режим праці та відпочинку – тривала безперервна робота за комп'ютером без перерв призводить до кумулятивного накопичення стомлення, що різко знижує ефективність і підвищує ризик помилок та травм.

5.2 Комплекс практичних рішень для мінімізації негативного впливу комп'ютеризованої роботи на здоров'я працівника

Для мінімізації вищезазначених ризиків необхідний комплексний підхід, що включає технічні, організаційні та індивідуальні заходи.

Ергономічна організація робочого місця:

– Робочий стіл та стілець – стілець повинен бути регульованим за висотою та глибиною сидіння, мати підтримку попереку (поясничний валик) та підлокітники. Висота робочої поверхні стола повинна забезпечувати положення, при якому передпліччя працівника знаходяться приблизно паралельно підлозі, а кути в ліктевих суглобах складають 70-90°.

– Розташування монітора – верхній край екрана повинен знаходитися на рівні очей або трохи нижче, на відстані 50-70 см від користувача. Для запобігання блисків екран слід розмістити перпендикулярно до вікна, а джерела загального освітлення – поза полем зору.

Оптимізація умов навколишнього середовища:

– Освітлення – слід забезпечити комбіноване освітлення: достатнє рівномірне загальне освітлення (не менше 300-500 лк) та м'яке, спрямоване місцеве світло для роботи з документами. Необхідно використовувати антиблікові фільтри на моніторах та регулювати яскравість і контрастність екрана під рівень освітлення в приміщенні.

– Мікроклімат – температура повітря має підтримуватися в межах 22-24°С, відносна вологість – 40-60%. Необхідно регулярне провітрювання приміщення та, за можливості, використання зволожувачів повітря для запобігання синдрому "сухого ока".

– Шум – рівень шуму не повинен перевищувати 50-60 дБА. Досягається це за рахунок використання тихого обладнання, звукоізоляції серверних кімнат та правильного планування офісного простору.

Організація режиму праці та відпочинку:

– Регламентовані перерви – при роботі за комп'ютером понад 50% робочого часу обов'язкові перерви тривалістю 10-15 хвилин кожні 45-60 хвилин роботи. Під час перерв необхідно відвернутися від екрана, зробити легкі фізичні вправи для очей (подивитися вдаль, обертати очима), спини та кистей рук.

– Гімнастика для очей – рекомендується виконувати короткі вправи кожні 20-30 хвилин: інтенсивно закривати і відкривати очі, обертати очима по колу, переводити погляд з близького предмета на далекий.

Індивідуальні засоби захисту та профілактика:

– Корекція зору – працівники з порушенням зору повинні використовувати окуляри зі спеціальним антирефлексним покриттям, призначеним для роботи за комп'ютером.

– Фізична активність – для профілактики скелетно-м'язових розладів важливо займатися фізичною культурою, віддавати перевагу активному відпочинку, а також виконувати короткі розминки протягом робочого дня.

Реалізація цих заходів не лише знизить ризик професійних захворювань і травматизму серед програмістів, але й створить комфортні умови для підвищення творчої продуктивності та якості виконуваної роботи, що є прямим інвестиційним внеском у людський капітал компанії.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Сучасне суспільство усвідомлює необхідність відповідального ставлення до екології, що стосується також сфери інформаційних технологій. Хоча процес розробки програмних систем сам по собі не супроводжується прямолінійними викидами в атмосферу або забрудненням водних ресурсів, існує низка опосередкованих, але значних впливів на навколишнє середовище, пов’язаних з життєвим циклом комп’ютерної техніки, енергоспоживанням та організацією офісного простору. Цей розділ присвячено ідентифікації основних екологічних аспектів діяльності ІТ-компанії та опису заходів, спрямованих на зменшення її екологічного сліду, що є важливою складовою корпоративної соціальної відповідальності.

6.1 Екологічні аспекти діяльності ІТ-компанії та пов’язані з ними ризики

Основним джерелом екологічного впливу в контексті розробки ПЗ є експлуатація інфраструктури даних центрів та офісної техніки. Ключовими негативними факторами вважаються:

– Підвищена енергоємність – робота серверів, комп'ютерних станцій, систем охолодження та освітлення офісних приміщень потребує значної кількості електроенергії. Виробництво цієї енергії, особливо якщо воно засноване на викопному паливі (вугіллі, газі), безпосередньо пов’язане з викидом парникових газів (CO_2), що сприяють глобальним кліматичним змінам. Також виділяються шкідливі речовини, такі як оксиди сірки та азоту.

– Утворення електронних відходів (е-відходів) – експлуатаційний термін комп'ютерного обладнання, моніторів, периферійних пристроїв та засобів зв'язку обмежений. Їхнє заміна генерує величезну кількість відходів, що містять небезпечні компоненти: свинець, ртуть, кадмій, полівінілхлорид (ПВХ). Неправильна утилізація таких відходів (наприклад, на звичайних звалищах або шляхом

примітивного спалювання) призводить до забруднення ґрунтів, грунтових вод та атмосфери токсичними речовинами, що становить загрозу для екосистем та здоров’я людей.

– Використання матеріалів та ресурсів – виробництво нової техніки вимагає видобутку рідкісних металів (золота, кобальту, танталу), використання пластиків (на основі нафтопродуктів) та великих обсягів прісної води. Це веде до виснаження природних ресурсів, деградації ландшафтів у районах видобутку та забруднення водних об’єктів.

– Забруднення повітряного середовища приміщень (мікроклімат) – хоча це стосується переважно охорони праці, погана вентиляція та використання матеріалів з виділенням летких органічних сполук (з меблів, килимів, офісної техніки) також формують несприятливе середовище як для працівників, так і для навколишнього середовища в цілому через підвищені вимоги до енергоспоживання систем кондиціонування..

6.2 Стратегії та практики для зменшення екологічного навантаження IT-сектора

Ефективне управління екологічними ризиками потребує впровадження проактивних стратегій на рівні компанії.

Політика енергоефективності та використання відновлюваної енергії

– Оптимізація парку техніки – перехід на енергоефективне обладнання (з сертифікатами Energy Star), централізація обчислювальних потужностей у віртуалізованих середовищах або хмарних сервісах, що дозволяє краще утилізувати ресурси.

– Керування енергоспоживанням – впровадження автоматичного відключення обладнання в нерабочий час, використання LED-освітлення з датчиками руху, оптимізація режимів роботи систем охолодження.

– «Зелена» енергія – закупівля електроенергії з відновлюваних джерел (сонячні, вітрові електростанції) або інвестування у власні відновлювані потужності.

Відповідальне поводження з відходами та принципи циркулярної економіки

– Пріоритет повторного використання та ремонту – продовження життєвого циклу обладнання через його модернізацію, ремонт та внутрішнє перерозподіл між підрозділами перед утилізацією.

– Контрольована утилізація е-відходів – обов'язкова передача застарілої техніки сертифікованим переробникам, які можуть безпечно виділити та переробити цінні (метали) і небезпечні компоненти, запобігаючи їх потраплянню на сміттєзвалища.

– Еко-закупівлі – надання пріоритету при закупівлях техніки, що має тривалий термін служби, можливість оновлення, а також вироблена з використанням вторинних матеріалів і легко розбирається для утилізації.

Зменшення споживання ресурсів та покращення культури виробництва

– Дематеріалізація процесів – максимальний перехід на безпаперовий документообіг, використання електронних підписів, цифрових архівів, що зменшує споживання паперу, чорнила та витрати на логістику.

– Відповідальність постачальників –включення екологічних критеріїв у вибір постачальників та партнерів, підтримка тих, хто дотримується стандартів екологічного менеджменту (ISO 14001).

– Екологічна свідомість персоналу – проведення навчальних заходів для формування екологічної культури серед співробітників, популяризація практик енергозбереження, роздільний збір сміття в офісі та інших ініціатив.

Реалізація зазначених підходів дозволяє ІТ-компанії не лише мінімізувати свій негативний вплив на природу, але й підвищити ефективність власних бізнес-процесів, покращити імідж на ринку та внести вагомий внесок у сталий розвиток суспільства.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було успішно вирішено завдання створення програмної системи для прогнозування розміру ПС, розроблених з використанням фреймворку Yii. Робота включала теоретичне дослідження, математичне моделювання, програмну реалізацію та економічне обґрунтування.

Основним науково-практичним результатом є побудована нелінійна регресійна модель для прогнозування розміру (LOC), що базується на архітектурних метриках (кількість класів, середня кількість методів, глибина дерева успадкування). Модель розроблена з урахуванням специфіки фреймворку Yii і продемонструвала високу точність (R^2=0.968, MMRE=0.131, PRED(0.25)=0.833), значно перевершуючи за ефективністю універсальні моделі та моделі, створені для інших PHP-фреймворків.

На основі цієї моделі було розроблено програмну систему «YiiSizeEstimator». Система реалізує автоматизований процес прогнозування, надаючи не лише точкову оцінку розміру, але й довірчі інтервали та інтервали прогнозування. Розробка велась з використанням об’єктно-орієнтованої методології та сучасного технологічного стеку (Python, NumPy, SciPy), що забезпечило модульність, надійність та зручність подальшого розвитку продукту. Результати комплексного тестування підтвердили повну відповідність системи висунутим функціональним вимогам.

Економічне обґрунтування доводить високу практичну цінність проєкту: розрахунковий термін окупності інвестицій у розробку становить лише 4 місяці завдяки потенційній економії коштів на плануванні та оптимізації використання ресурсів. Також у роботі розглянуто важливі аспекти безпечної та екологічно відповідальної експлуатації системи у розділах охорони праці та охорони навколишнього середовища.

Таким чином, мета роботи досягнута: створено науково обґрунтований, технологічно реалізований та економічно ефективний інструмент, що дозволяє

суттєво підвищити достовірність і об’єктивність раннього прогнозування розміру ПС на фреймворку Yii, що є особливо актуальним в умовах переходу на нову мажорну версію Yii 3.0.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chiyangwa T. B., Mnkandla E. Modelling the critical success factors of agile software development projects in South Africa. *SA Journal of Information Management*. 2017. Vol. 19, no. 1. URL: https://doi.org/10.4102/sajim.v19i1.838 (date of access: 20.11.2025).

2. Boehm, B. W. (2000). *Software cost estimation with Cocomo II*. Prentice Hall.

3. *The State of PHP 2025 – expert review*. The JetBrains Blog. (2025, December 9). https://blog.jetbrains.com/phpstorm/2025/10/state-of-php-2025/.

4. Kiewkanya, M., & Surak, S. (2016). Constructing C++ software size estimation model from class diagram. *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 1–6. https://doi.org/10.1109/jcsse.2016.7748880.

5. Prykhodko, N. V., & Prykhodko, S. B. (2018a). Non-linear regression model to estimate the software size of VB-based information systems. *Information Technology And Computer Engineering, 43(3)*, 37–42. https://doi.org/10.31649/1999-9941-2018-43-3-37-42.

6. Prykhodko, N. V., & Prykhodko, S. B. (2018). The non-linear regression model to estimate the software size of open source Java-based systems. *Radio Electronics, Computer Science, Control*, 0(3). https://doi.org/10.15588/1607-3274-2018-3-17.

7. Prykhodko, S. B., Prykhodko, N. V., Farionova, T. A., & Vorona, M. V. (2020). Three-factor non-linear regression model to estimate the size of open source php-based applications. *Scientific Notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences*, 1(1), 124–131. https://doi.org/10.32838/2663-5941/2020.1-1/23

8. Prykhodko, S. B., Shutko, I. S., & Prykhodko, A. S. (2022). A nonlinear regression model to estimate the size of web apps created using the CAKEPHP framework. *Radio Electronics, Computer Science, Control*, (4), 129–139. https://doi.org/10.15588/1607-3274-2021-4-12.

9. Приходько С.Б., Приходько Н.В. Нелінійне регресійне рівняння для оцінювання розміру програмного забезпечення інформаційних систем з відкритим кодом на PHP. *Вісник східноукраїнського національного університету імені Володимира Даля.* 2018. № 6 (247). С. 135–139.

10. Prykhodko, S., Shutko, I., & Prykhodko, A. (2022). Early size estimation of web apps created using codeigniter framework by nonlinear Regression Models. *RADIOELECTRONIC AND COMPUTER SYSTEMS*, (3), 84–94. https://doi.org/10.32620/reks.2022.3.06

11. Приходько С. Б., Приходько Н. В., Ворона М. В., Бєловол І. О. Нелінійна регресійна модель для оцінювання розміру Web-застосунків, що створюються з використанням фреймворку Laravel. *Інформаційні технології та комп'ютерна інженерія.* 2021. Т. 50, № 1. С. 115–121.

12. Fenton N., Bieman J. Software Metrics. CRC Press, 2014. URL: https://doi.org/10.1201/b17461 (date of access: 20.11.2025).

13. Prykhodko, S. B., Prykhodko, N. V., & Koltsov, A. V. (2024). A nonlinear regression model for early LOC estimation of open-source Kotlin-based applications. *Radio Electronics, Computer Science, Control*, (1), 85. https://doi.org/10.15588/1607-3274-2024-1-8.

14. LATANSKA, L., MAKAROVA, L., KOLTSOV, A., & DAVLATOVA, D. (2022). A nonlinear regression model for estimating the size of web applications created using SYMFONY framework. *Herald of Khmelnytskyi National University. Technical Sciences*, *315*(6(1)), 119–124. https://doi.org/10.31891/2307-5732-2022-315-6-119-124.

15. Каіров В.О., Смітієнко В.О. РАННЄ ОЦІНЮВАННЯ РОЗМІРУ ВЕБЗАСТОСУНКІВ, РОЗРОБЛЕНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ YII. *Інформаційні технології: моделі, алгоритми, системи* : зб. тез V всеукраїнської науково-практичної інтернет конференції, 16-17 листопада 2025 р. С 64–65.

16. *Expert judgment in Project Management*. Galorath. (2025, November 20). https://galorath.com/estimation/expert-judgment/.

17. Sebastian. (2023a, August 19). *Analogous estimating: Definition, examples, pros & cons*. Project. https://project-management.info/analogous-estimating/.

18. Sebastian. (2023b, August 19). *Bottom-up estimating – definition, example, Pros & Cons*. Project. https://project-management.info/bottom-up-estimating-definition-example-pros-cons/.

19. Sebastian. (2023, August 19). *Three-point estimating and Pert Distribution (Cost & time estimation)*. Project. https://project-management.info/three-point-estimating-pert/.

20. Cohn, M. (n.d.). *Estimating with use case points*. Mountain Goat Software. https://www.mountaingoatsoftware.com/articles/estimating-with-use-case-points.

21. Maxym, Z. (2023, May 27). *Широкосмуговий метод оцінки Дельфі (wideband Delphi)*. Maxym Zosym. https://www.maxzosim.com/wideband-delphi/.

22. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6), 476–493. https://doi.org/10.1109/32.295895.

23. *How about progress on yii3 development?: Yii PHP framework*. Yii Framework. (n.d.). https://www.yiiframework.com/yii3-progress.

24. Prykhodko, S., Shutko, I., & Prykhodko, A. (2021). Early LOC estimation of web apps created using YII framework by nonlinear Regression Models. WSEAS TRANSACTIONS ON COMPUTERS, 20, 321–328. https://doi.org/10.37394/23205.2021.20.35.

25. Frost J. Regression Analysis: An Intuitive Guide for Using and Interpreting Linear Models. Statistics By Jim Publishing, 2020. 355 p.

26. Приходько С. Б., Макарова Л. М., Приходько Н. В., Пухалевич А. В. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни «Емпіричні методи програмної інженерії». Миколаїв : НУК, 2023. 56 с.

27. Prykhodko, S., Prykhodko, N., Makarova, L., & Pugachenko, K. (2017). Detecting outliers in multivariate non-Gaussian data on the basis of normalizing transformations. *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, 846–849. https://doi.org/10.1109/ukrcon.2017.8100366

28. Prykhodko, S., Prykhodko, N., Makarova, L., & Pukhalevych, A. (2018). Application of the squared mahalanobis distance for detecting outliers in multivariate

non-Gaussian Data. *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, 962–965. https://doi.org/10.1109/tcset.2018.8336353

29. PRYKHODKO, N. V., & PRYKHODKO, S. B. (2018). Constructing the nonlinear regression models on the basis of multivariate normalizing transformations. *Èlektronnoe Modelirovanie, 40(6)*, 101–110. https://doi.org/10.15407/emodel.40.06.101

30. Prykhodko, S., & Prykhodko, N. (2020). Mathematical modeling of Non-Gaussian dependent random variables by nonlinear regression models based on the multivariate normalizing transformations. *Advances in Intelligent Systems and Computing*, 166–174. https://doi.org/10.1007/978-3-030-58124-4_16.

31. Foss T., Stensrud E., Kitchenham B., Myrtveit I. A Simulation Study of the Model Evaluation Criterion MMRE // *IEEE Transactions on Software Engineering.* – 2003. – Vol. 29, No. 11. – P. 985–995.300.

32. Prykhodko, S., & Prykhodko, N. (2020a). Mathematical modeling of Non-Gaussian dependent random variables by nonlinear regression models based on the multivariate normalizing transformations. *Advances in Intelligent Systems and Computing*, 166–174. https://doi.org/10.1007/978-3-030-58124-4_16.

33. Booch, G., Rumbaugh, J., & Jacobson, I. (2015). *The Unified Modeling Language User Guide: Covers UML 2.0.* Pearson Education.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Найменування системи: «YiiSizeEstimator».

1 Підстави для розробки

Підставою для розробки є тема кваліфікаційної магістерської роботи та наказ на її затвердження.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням системи є надання користувачеві можливості здійснювати достовірне прогнозування розміру (у рядках коду – LOC) ПС, розроблених з використанням фреймворку Yii, на основі побудованої нелінійної регресійної моделі.

2.2 Експлуатаційне призначення

Система призначена для експлуатації на персональних комп’ютерах та ноутбуках керівників проєктів, архітекторів та розробників для раннього прогнозування обсягу коду на етапі проєктування архітектури.

3 Вимоги до програми або програмного продукту

3.1 Вимоги для функціональних характеристик

3.1.1 Вимоги до переліку виконуваних функцій

Розроблена система має надавати можливість виконувати наступні функції:

– Прогнозування очікуваного розміру ПС (у LOC) на основі введених архітектурних метрик.

– Розрахунок довірчого інтервалу для отриманого значення прогнозу.

– Розрахунок інтервалу прогнозування для отриманого значення прогнозу.

3.1.2 Вимоги для організації вхідних та вихідних даних

Введення вхідних даних відбувається через графічний інтерфейс користувача. Результати виводяться у відповідні поля інтерфейсу.

Вхідними даними для проведення прогнозу є:

1) Параметри попередньо навченої моделі (параметри λ_Y, λ_X_1, λ_X_2, λ_X_3).

2) Значення трьох метрик для оцінюваної ПС: Кількість класів (Classes, X_1): ціле число, більше 0. Середня кількість методів на клас (Avg num of methods, X_2): дійсне додатне число. Глибина дерева успадкування (DIT, X_3): дійсне невід'ємне число (0 або більше).

3) Рівень довіри (довірча ймовірність): дійсне число в діапазоні (0, 1), наприклад, 0.95.

Вхідними даними для проведення прогнозу є:

1) Точковий прогноз розміру ПС (очікувана кількість рядків коду – LOC).

2) Нижня та верхня межі довірчого інтервалу.

3) Нижня та верхня межі інтервалу прогнозування.

Усі значення подаються у числовому вигляді.

3.2 Вимоги до надійності

Стійке функціонування має бути забезпечене реалізацією валідації всіх вхідних даних на коректність типу, діапазону та повноти. У разі виявлення некоректних даних система повинна надавати зрозуміле повідомлення про помилку з описом проблеми.

3.3 Умови експлуатації

Кліматичні умови експлуатації повинні відповідати вимогам, встановленим для побутової електронної та обчислювальної техніки.

3.4 Вимоги до технічної та програмної сумісності

Мінімальні апаратні вимоги: 64-бітний процесор архітектури x86-64, 512 МБ оперативної пам’яті, 150 МБ вільного місця на диску. Необхідне програмне забезпечення: операційна система Windows (7 та новіші), Linux (дистрибутиви на основі glibc, такі як Ubuntu 18.04+) або macOS, а також встановлений інтерпретатор Python версії 3.8 або вище.

3.5 Вимоги до захисту інформації та програм

Спеціальні вимоги до захисту інформації не висуваються.

3.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки не висуваються, оскільки розповсюдження передбачається в електронному вигляді.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4 Вимоги до програмної документації

Склад програмної документації має включати:

1) Технічне завдання (цей документ);
2) Програму та методику випробувань;
3) Інструкцію користувача (User Guide);
4) Опис програмного забезпечення (Software Description);
5) Вихідний текст програми (Source Code).

5 Техніко-економічні показники

Економічна ефективність від розробки та впровадження програмної системи розрахована в розділі 4 роботи. Відповідно до результатів розрахунків, впровадження є доцільним, а термін окупності інвестицій становить приблизно 4 місяці.

6 Стадії та етапи розробки програмної системи забезпечення

Стадії та етапи виконання розробки програмної системи представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

Стадії розробки	Назва етапу	Термін виконання
Технічне завдання	Розробка технічного завдання	25.09.2025
	Затвердження технічного завдання	07.10.2025
Загальносистемні рішення	Вибір засобів моделювання (UML)	19.10.2025
	Розробка та специфікація варіантів використання	27.10.2025
	Розробка діаграми діяльності системи	14.11.2025
Рішення з інформаційного забезпечення	Розробка рішення з інформаційного забезпечення	25.11.2025
Рішення з програмного забезпечення	Вибір засобів розробки	27.11.2025
	Моделювання статичної структури системи	29.11.2025
	Моделювання взаємодії системи	02.12.2025
	Тестування та випробування системи	06.12.2025

7 Порядок контролю та прийомки

Приймально-здавальні випробування проводяться Замовником або уповноваженою ним особою на основі документа «Програма та методика випробувань». Результати випробувань фіксуються у «Протоколі проведення випробувань». Підставою для закінчення робіт є підписання сторонами «Акту прийому-здачі системи у експлуатацію».

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програмна система «YiiSizeEstimator», призначена для прогнозування розміру ПС, розроблених з використанням фреймворку Yii.

2 Мета випробувань

Метою випробувань є верифікація працездатності системи та перевірка відповідності її фактичних функціональних характеристик вимогам, викладеним у технічному завданні та основній частині роботи.

3 Вимоги до застосунку

У ході випробувань здійснюється перевірка функціональних характеристик, визначених у технічному завданні. Розроблена система має забезпечувати виконання наступних функцій:

– Прогнозування очікуваного розміру ПС (у рядках коду – LOC) на основі введених метрик.

– Розрахунок довірчого інтервалу для отриманого прогнозу.

– Розрахунок інтервалу прогнозування для отриманого прогнозу.

– Коректна обробка вхідних даних, включаючи валідацію та обробку помилок.

4 Вимоги до програмної документації

Перелік необхідної програмної документації включає:

1) технічне завдання;

2) програму та методику випробувань;

3) інструкцію користувача;

4) опис програмного забезпечення.

5) текст програми

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Для проведення випробувань використовувався персональний комп’ютер з наступною конфігурацією:

– Модель: Ноутбук на базі Intel Core Ultra 5 226V.

– Екран: 14" OLED, глянцевий, 2880x1800 (2.8K), 120 Гц.

– Процесор: Intel Core Ultra 5 226V (15-е покоління Lunar Lake, 8 ядер, 2.1 - 4.5 ГГц) з нейропроцесором Intel AI Boost NPU.

– Графіка: Інтегрована Intel Arc Graphics 130V.

– Оперативна пам'ять: 16 ГБ DDR5.

– Накопичувач: SSD 512 ГБ

5.2 Програмні засоби, що використовувались під час випробувань

– Операційна система: Windows 11 Pro 23H2 (зборка 22631.3447).

– Середовище виконання: Python 3.11.5.

5.3 Порядок проведення випробувань

Порядок проведення випробувань передбачає послідовну перевірку відповідності функціональних характеристик системи заявленим вимогам, а також перевірку наявності та оформлення всієї необхідної програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

За відсутності специфічних вимог до формату документів, наявність та відповідність програмної документації зазначеному вище переліку перевіряється візуально представником замовника або уповноваженою особою.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмної системи

Перевірка працездатності реалізована відповідно до розділу «Вимоги до функціональних характеристик» технічного завдання. Випробування вважаються успішними при повній відповідності фактичної поведінки системи очікуваній.

Порядок випробувань включає:

– Перевірку коректності роботи графічного інтерфейсу.

– Тестування валідації вхідних даних: введення коректних значень, некоректних типів даних, значень за межами допустимих діапазонів.

– Перевірку основної функціональності: введення набору тестових метрик проєкту та параметрів моделі з подальшою перевіркою коректності отриманих результатів (точкового прогнозу, меж довірчого інтервалу та інтервалу прогнозування).

7 Результати випробувань

За результатами проведених випробувань було підтверджено, що програмна система «YiiSizeEstimator» повною мірою відповідає всім висунутим функціональним вимогам..

8 Відомості про відмови, збої та аварійні ситуації

Протягом усього процесу тестування не було зафіксовано жодних відмов у роботі системи, збоїв, критичних помилок чи аварійних ситуацій, що призводять до неконтрольованого завершення роботи.

9 Відомості про коригування параметрів об’єкта випробувань та технічної документації

Під час виконання випробувань не проводилося коригування параметрів об’єкта тестування (програмного коду) та технічної документації.

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА

Вступ

Ця інструкція призначена для ознайомлення з функціоналом та порядком роботи з програмною системою «YiiSizeEstimator». У ній описуються основні елементи інтерфейсу та кроки, необхідні для отримання прогнозу розміру ПС.

Загальні відомості про систему

Найменування системи: «YiiSizeEstimator».

Використані мови програмування

Система розроблена мовою програмування Python. Для реалізації графічного інтерфейсу користувача використана бібліотека Qt.

Призначення системи

Система призначена для прогнозування обсягу коду (у рядках коду – LOC) ПС, розроблених з використанням фреймворку Yii. Прогноз ґрунтується на аналізі трьох ключових архітектурних метрик: кількість класів, середня кількість методів на клас та глибина дерева успадкування.

Функціональні можливості системи

– Точкове прогнозування очікуваного розміру ПС (LOC).

– Розрахунок довірчого інтервалу для отриманого прогнозу.

– Розрахунок інтервалу прогнозування для конкретної оцінюваної ПС.

– Валідація вхідних даних з наданням зрозумілих повідомлень про помилки.

Обмеження області застосування системи

Область застосування системи обмежується процесом планування та управління проєктами розробки програмних систем, зокрема для систем, що використовують фреймворк Yii

Умови, необхідні для використання системи

Спеціальні умови відсутні.

Вимоги до технічної сумісності та програмного середовища

Апаратне забезпечення: 64-бітний процесор (архітектури x86-64), не менше

512 МБ оперативної пам'яті, приблизно 150 МБ вільного місця на диску.

Програмне забезпечення: Операційна система Windows (7/8/10/11), Linux (сучасні дистрибутиви, наприклад, Ubuntu 20.04+) або macOS. Обов’язково має бути встановлений інтерпретатор Python версії 3.8 або вище.

Опис роботи з програмою

Графічний інтерфейс програмної системи «YiiSizeEstimator» містить наступні основні елементи для введення даних та відображення результатів (загальний вигляд наведено на рисунку В.1):

– Поле «Кількість класів (X_1)» – для введення цілого додатного числа, що відповідає загальній кількості класів в оцінюваному проєкті.

– Поле «Середня кількість методів (X_2)» – для введення додатного дійсного числа, яке представляє середню кількість методів, що припадає на один клас.

– Поле «Глибина успадкування (X_3)» – для введення невід'ємного дійсного числа, що визначає середню глибину дерева успадкування в проєкті.

– Поля для параметрів λ (лямбда):

1) «λ для Y (λ_Y)» – параметр нормалізуючого перетворення для залежної змінної.

2) «λ для X_1 (λ_X_1)» – параметр перетворення для кількості класів.

3) «λ для X_2 (λ_X_2)» – параметр перетворення для середньої кількості методів.

4) «λ для X_3 (λ_X_3)» – параметр перетворення для глибини успадкування. (Ці параметри є частиною математичної моделі та надаються разом із системою).

– Поле «Рівень довіри» – для вказання довірчої ймовірності (наприклад, 0.95 або 95%) у вигляді числа в діапазоні від 0 до 1.

– Кнопка «Розрахувати» – запускає процес обчислення за введеними даними.

– Область виведення результатів – текстова область або окремі поля, де відображаються результати розрахунків після натискання кнопки.

Порядок отримання прогнозу:

– Ввести значення трьох метрик проєкту у відповідні текстові поля (X_1, X_2, X_3).

– Ввести значення параметрів λ (λ_Y, λ_X_1, λ_X_2, λ_X_3), отримані для

попередньо навченої математичної моделі.

– Ввести бажаний рівень довіри (за замовчуванням може бути 0.95).

– Натиснути кнопку «Розрахувати»

У разі введення невалідних вхідних даних (наприклад, від’ємних чисел або текстових значень) система виведе повідомлення про помилку із закликом виправити введену інформацію.

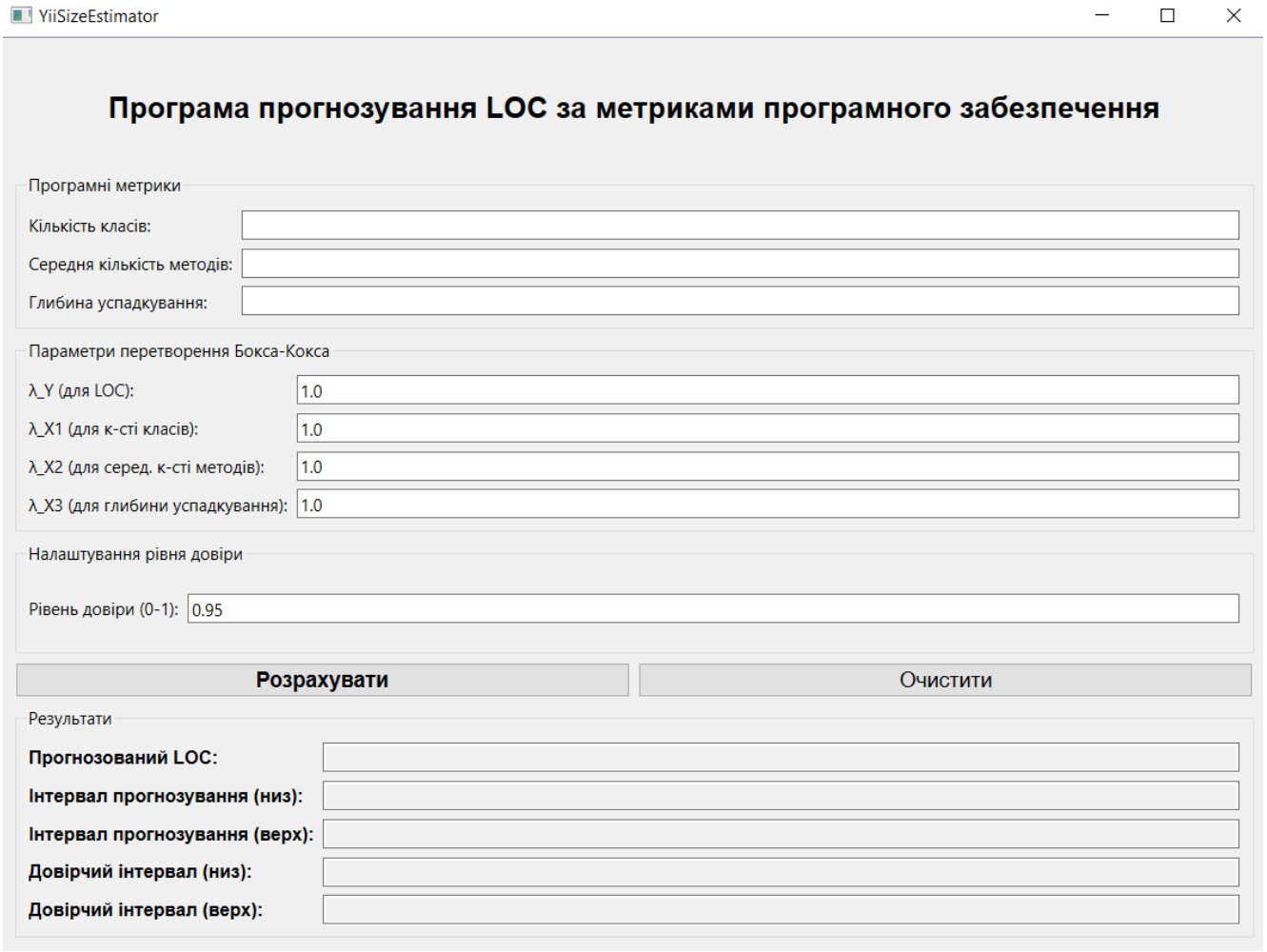

Рисунок В.1 – Запуск системи

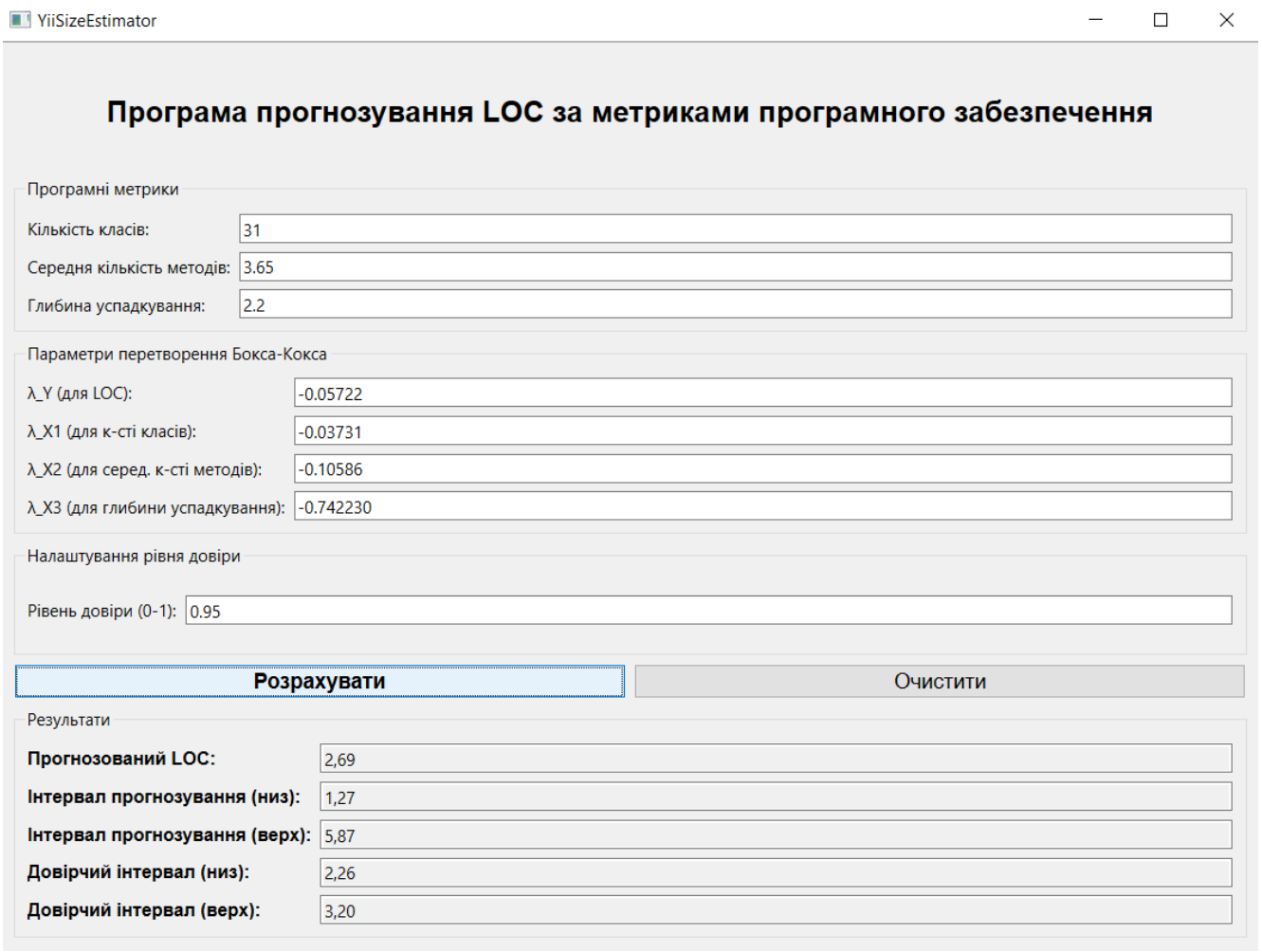

Рисунок В.2 – Результат роботи системи

ДОДАТОК Г – ТЕКСТ ПРОГРАМИ

```
import numpy as np
import pandas as pd
import math
from scipy.stats import t

class Evaluator:
    def __init__(self):
        # Pre-calculated model parameters as constants
        self.regression_coefficients = np.array([1.05416, 1.39398, 0.51635])
        self.intercept = -4.43109

        # Mean values of transformed variables
        self.column_means = np.array([4.0272, 1.4126, 0.4560])

        # Inverse covariance matrix
        self.inverse_cov_matrix = np.array([
            [0.0205, 0.0234, 0.0471],
            [0.0234, 0.2263, -0.0114],
            [0.0471, -0.0114, 0.7125]
        ])

        # Standard error
        self.error_standard = 0.3487

        # Training data size
        self.total_data_size = 48

        # Number of features
        self.features_count = 3

    def box_cox_transform(self, x, lambda_param):
        """Apply Box-Cox transformation with given lambda parameter"""
        if lambda_param != 0:
            return (x ** lambda_param - 1) / lambda_param
        else:
            return np.log(x)

    def inverse_box_cox_transform(self, z, lambda_param):
        """Inverse Box-Cox transformation"""
        if lambda_param != 0:
            return (lambda_param * z + 1) ** (1 / lambda_param)
        else:
            return np.exp(z)

    def project_data_metrics(self, metrics_data, lambdas, confidence_level=0.95):
```

```
        """
        Project metrics using Box-Cox transformation

        Parameters:
        -----------
        metrics_data : DataFrame with columns X1, X2, X3
        lambdas : list of 4 lambda values [lambda_X1, lambda_X2, lambda_X3, lambda_Y]
        confidence_level : float, confidence level (default 0.95)

        Returns:
        --------
        DataFrame with prediction and intervals
        """
        # Extract lambda values
        lambda_X1, lambda_X2, lambda_X3, lambda_Y = lambdas

        # Apply Box-Cox transformation to input metrics
        Z_X1 = self.box_cox_transform(metrics_data['X1'].values[0], lambda_X1)
        Z_X2 = self.box_cox_transform(metrics_data['X2'].values[0], lambda_X2)
        Z_X3 = self.box_cox_transform(metrics_data['X3'].values[0], lambda_X3)

        transformed_values = np.array([Z_X1, Z_X2, Z_X3])

        # Calculate prediction in transformed space
        # Y = b0 + b1*Z_X1 + b2*Z_X2 + b3*Z_X3
        z_pred = self.intercept + np.dot(self.regression_coefficients,
transformed_values)

        # Calculate centered values (deviation from means)
        centered_values = transformed_values - self.column_means

        # Calculate covariance scalar: (Z - Z_mean)^T * Cov^-1 * (Z - Z_mean)
        covariance_scalar = np.dot(np.dot(centered_values, self.inverse_cov_matrix),
centered_values.T)

        # Calculate t-quantile for confidence intervals
        alpha = 1 - confidence_level
        df = self.total_data_size - self.features_count - 1  # degrees of freedom
        t_quantile = t.ppf(1 - alpha / 2, df)

        # Calculate margins of error
        # For confidence interval of mean prediction
        margin_error = t_quantile * self.error_standard * math.sqrt(
            1 / self.total_data_size + covariance_scalar
        )

        # For prediction interval of individual prediction
        prediction_margin = t_quantile * self.error_standard * math.sqrt(
            1 + (1 / self.total_data_size) + covariance_scalar
        )
```

```
        # Transform predictions back to original scale using inverse Box-Cox
        prediction_original = self.inverse_box_cox_transform(z_pred, lambda_Y)
        lower_conf = self.inverse_box_cox_transform(z_pred - margin_error, lambda_Y)
        upper_conf = self.inverse_box_cox_transform(z_pred + margin_error, lambda_Y)
        lower_pred = self.inverse_box_cox_transform(z_pred - prediction_margin,
lambda_Y)
        upper_pred = self.inverse_box_cox_transform(z_pred + prediction_margin,
lambda_Y)

        # Create result dictionary
        result = {
            'prediction_regression': [prediction_original],
            'lower_bound_confidence': [lower_conf],
            'upper_bound_confidence': [upper_conf],
            'lower_bound_prediction': [lower_pred],
            'upper_bound_prediction': [upper_pred],
            'z_prediction': [z_pred],
            'covariance_scalar': [covariance_scalar]
        }

        return pd.DataFrame(result)



import sys
from PyQt6.QtWidgets import (QApplication, QMainWindow, QWidget, QVBoxLayout,
QHBoxLayout, QGridLayout,
                             QLabel, QLineEdit, QPushButton, QGroupBox, QScrollArea,
QMessageBox)
from PyQt6.QtCore import Qt, QLocale
from PyQt6.QtGui import QFont, QDoubleValidator, QIntValidator
import pandas as pd
from Evaluator import Evaluator

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("YiiSizeEstimator")
        self.resize(900, 650)
        self.setupUi()

    def setupUi(self):
        # Scrollable central widget
        centralWidget = QWidget()
        scrollArea = QScrollArea()
        scrollArea.setWidgetResizable(True)
        scrollArea.setWidget(centralWidget)
        self.setCentralWidget(scrollArea)
```

```
        # Main Layout
        mainLayout = QVBoxLayout()
        centralWidget.setLayout(mainLayout)

        # Header
        headerLabel = QLabel("Програма оцінки LOC за метриками програмного
забезпечення")
        headerLabel.setFont(QFont("Arial", 16, QFont.Weight.Bold))
        headerLabel.setAlignment(Qt.AlignmentFlag.AlignCenter)
        mainLayout.addWidget(headerLabel)

        # Input Section - Software Metrics
        metricsGroupBox = QGroupBox("Програмні метрики")
        metricsGridLayout = QGridLayout()
        metricsGroupBox.setLayout(metricsGridLayout)
        mainLayout.addWidget(metricsGroupBox)

        self.classCountInput = self.createTextInput("Кількість класів:",
                                                     QIntValidator(1, 999999),
                                                     metricsGridLayout, 0)
        self.avgMethodsInput = self.createTextInput("Середня кількість методів:",
                                                     QDoubleValidator(0.01, 9999.0,
2),
                                                     metricsGridLayout, 1)
        self.ditValueInput = self.createTextInput("Глибина успадкування:",
                                                   QDoubleValidator(0.01, 9999.0, 2),
                                                   metricsGridLayout, 2)

        # Lambda Parameters Section
        lambdaGroupBox = QGroupBox("Параметри перетворення Бокса-Кокса")
        lambdaGridLayout = QGridLayout()
        lambdaGroupBox.setLayout(lambdaGridLayout)
        mainLayout.addWidget(lambdaGroupBox)

        doubleValidator = QDoubleValidator(-10.0, 10.0, 4)

        self.lambdaYInput = self.createTextInput("λ_Y (для LOC):",
                                                 doubleValidator,
                                                 lambdaGridLayout, 0, "0.1")
        self.lambdaX1Input = self.createTextInput("λ_X1 (для к-сті класів):",
                                                  doubleValidator,
                                                  lambdaGridLayout, 1, "0.1")
        self.lambdaX2Input = self.createTextInput("λ_X2 (для серед. к-сті методів):",
                                                  doubleValidator,
                                                  lambdaGridLayout, 2, "0.1")
        self.lambdaX3Input = self.createTextInput("λ_X3 (для глибини успадкування):",
                                                  doubleValidator,
                                                  lambdaGridLayout, 3, "0.1")

        # Confidence Level
```

```
        confidenceGroupBox = QGroupBox("Налаштування рівня довіри")
        confidenceLayout = QGridLayout()
        confidenceGroupBox.setLayout(confidenceLayout)
        mainLayout.addWidget(confidenceGroupBox)

        self.confidenceInput = self.createTextInput("Рівень довіри (0-1):",
                                                     QDoubleValidator(0.0, 1.0, 3),
                                                     confidenceLayout, 0, "0.95")

        # Buttons
        buttonLayout = QHBoxLayout()
        self.calculateBtn = QPushButton("Розрахувати")
        self.calculateBtn.setFont(QFont("Arial", 11, QFont.Weight.Bold))
        self.calculateBtn.clicked.connect(self.performCalculation)
        self.clearBtn = QPushButton("Очистити")
        self.clearBtn.setFont(QFont("Arial", 11))
        self.clearBtn.clicked.connect(self.resetFields)
        buttonLayout.addWidget(self.calculateBtn)
        buttonLayout.addWidget(self.clearBtn)
        mainLayout.addLayout(buttonLayout)

        # Output Section
        outputGroupBox = QGroupBox("Результати")
        outputGridLayout = QGridLayout()
        outputGroupBox.setLayout(outputGridLayout)
        mainLayout.addWidget(outputGroupBox)

        self.projectEstimateOutput = self.createOutputField("Прогнозований LOC:",
                                                             outputGridLayout, 0)
        self.lowerPredOutput = self.createOutputField("Інтервал прогнозування
(низ):",
                                                       outputGridLayout, 1)
        self.upperPredOutput = self.createOutputField("Інтервал прогнозування
(верх):",
                                                       outputGridLayout, 2)
        self.lowerConfOutput = self.createOutputField("Довірчий інтервал (низ):",
                                                       outputGridLayout, 3)
        self.upperConfOutput = self.createOutputField("Довірчий інтервал (верх):",
                                                       outputGridLayout, 4)

    def createTextInput(self, label, validator, layout, row, defaultValue=""):
        labelWidget = QLabel(label)
        textInput = QLineEdit()

        # Create a custom validator that accepts comma as decimal separator
        if isinstance(validator, QDoubleValidator):
            validator.setNotation(QDoubleValidator.Notation.StandardNotation)
            validator.setLocale(QLocale(QLocale.Language.English,
QLocale.Country.UnitedStates))
```

```
        textInput.setValidator(validator)
        textInput.setText(defaultValue)
        layout.addWidget(labelWidget, row, 0)
        layout.addWidget(textInput, row, 1)
        return textInput

    def createOutputField(self, label, layout, row):
        labelWidget = QLabel(label)
        labelWidget.setFont(QFont("Arial", 10, QFont.Weight.Bold))
        outputField = QLineEdit()
        outputField.setReadOnly(True)
        outputField.setStyleSheet("background-color: #f0f0f0;")
        layout.addWidget(labelWidget, row, 0)
        layout.addWidget(outputField, row, 1)
        return outputField

    def resetFields(self):
        self.classCountInput.clear()
        self.avgMethodsInput.clear()
        self.ditValueInput.clear()
        self.lambdaYInput.setText("1.0")
        self.lambdaX1Input.setText("1.0")
        self.lambdaX2Input.setText("1.0")
        self.lambdaX3Input.setText("1.0")
        self.confidenceInput.setText("0.95")
        self.projectEstimateOutput.clear()
        self.lowerPredOutput.clear()
        self.upperPredOutput.clear()
        self.lowerConfOutput.clear()
        self.upperConfOutput.clear()

    def performCalculation(self):
        try:
            # Helper function to convert comma to dot for float conversion
            def parse_float(text):
                return float(text.replace(',', '.'))

            # Get input values
            classCount = parse_float(self.classCountInput.text())
            avgMethods = parse_float(self.avgMethodsInput.text())
            ditValue = parse_float(self.ditValueInput.text())

            lambda_Y = parse_float(self.lambdaYInput.text())
            lambda_X1 = parse_float(self.lambdaX1Input.text())
            lambda_X2 = parse_float(self.lambdaX2Input.text())
            lambda_X3 = parse_float(self.lambdaX3Input.text())

            confidenceLevel = parse_float(self.confidenceInput.text())

            # Validate inputs
```

```
            if classCount <= 0 or avgMethods <= 0 or ditValue <= 0:
                raise ValueError("All metric values must be greater than 0.")

            if confidenceLevel <= 0 or confidenceLevel >= 1:
                raise ValueError("Confidence level must be between 0 and 1.")

            # Create lambdas array: [lambda_X1, lambda_X2, lambda_X3, lambda_Y]
            lambdas = [lambda_X1, lambda_X2, lambda_X3, lambda_Y]

            # Create handler with pre-calculated parameters
            handler = Evaluator()

            # Create input metrics dataframe
            metrics_df = pd.DataFrame({
                "X1": [classCount],
                "X2": [avgMethods],
                "X3": [ditValue]
            })

            # Get predictions
            result = handler.project_data_metrics(
                metrics_data=metrics_df,
                lambdas=lambdas,
                confidence_level=confidenceLevel
            )

            # Display results (format with comma as decimal separator)
            self.projectEstimateOutput.setText(f"{result['prediction_regression'][0]:
.2f}".replace('.', ','))
            self.lowerPredOutput.setText(f"{result['lower_bound_prediction'][0]:.2f}"
.replace('.', ','))
            self.upperPredOutput.setText(f"{result['upper_bound_prediction'][0]:.2f}"
.replace('.', ','))
            self.lowerConfOutput.setText(f"{result['lower_bound_confidence'][0]:.2f}"
.replace('.', ','))
            self.upperConfOutput.setText(f"{result['upper_bound_confidence'][0]:.2f}"
.replace('.', ','))

        except ValueError as ve:
            QMessageBox.warning(self, "Input Error", str(ve))
        except Exception as e:
            QMessageBox.critical(self, "Error", f"An error occurred: {str(e)}")

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

ДОДАТОК Д – ОПИС ПРОГРАМИ

1 Загальні відомості

Найменування: «YiiSizeEstimator».

1.2 Програмне забезпечення, необхідне для функціонування системи

Для коректної роботи програмної системи необхідне наступне ПЗ:

– Операційна система: Windows (7, 8, 10, 11), Linux (сучасні дистрибутиви, наприклад, Ubuntu 20.04+) або macOS.

– Середовище виконання – інтерпретатор Python версії 3.8 або новішої.

– Бібліотеки Python – стандартні бібліотеки, а також NumPy, SciPy. Для графічного інтерфейсу знадобиться PythonQt (входить до складу Python) або інший обраний фреймворк.

1.3 Мови програмування

Програмна система розроблена мовою програмування Python. Для виконання складних математичних обчислень, реалізації статистичних методів та нормалізуючих перетворень використовуються бібліотеки NumPy та SciPy. Графічний інтерфейс користувача реалізований за допомогою бібліотеки Python Qt.

2 Функціональне призначення

Функціональне призначення системи полягає у наданні інструменту для прогнозування розміру (у рядках коду – LOC) програмних систем, розроблених з використанням фреймворку Yii, на основі архітектурних метрик.

Доступні користувачеві функції:

– Точкове прогнозування очікуваного розміру коду ПС.

– Розрахунок довірчого інтервалу для отриманого прогнозу.

– Розрахунок інтервалу прогнозування для отриманого прогнозу.

– Валідація вхідних даних.

2.1 Функціональні обмеження

Система призначена виключно для прогнозування розміру ПС, архітектура яких відповідає моделі, закладеної в її математичне ядро (орієнтованої на Yii). Вона не виконує автоматичного збору метрик з вихідного коду.

3 Технічні засоби, що використовуються

Мінімальні апаратні вимоги для експлуатації:

– Процесор – 64-бітний (архітектура x86-64).

– Оперативна пам’ять – 512 МБ.

– Вільне місце на диску – 150 МБ.

4 Виклик та завантаження

Система запускається виконанням головного скрипту (yii_size_estimator.py) через інтерпретатор Python або клацанням на ярлик виконуваного файлу, якщо система була поставлена у самостійний застосунок (наприклад, за допомогою PyInstaller).

5 Вхідні дані

Вхідні дані вводяться користувачем вручну через інтерфейс системи. Вхідними даними для системи є:

1) Значення трьох метрик проєкту:

– Кількість класів (X_1) – ціле додатне число (більше 0).

– Середня кількість методів на клас (X_2) – дійсне додатне число.

– Глибина дерева успадкування (X_3) – дійсне невід’ємне число.

2) Параметри математичної моделі (λ):

– λ_Y – параметр нормалізації для залежної змінної (розміру).

– λ_X_1 – параметр нормалізації для кількості класів.

– λ_X_2 – параметр нормалізації для середньої кількості методів.

– λ_X_3 – параметр нормалізації для глибини успадкування. (Ці чотири значення є константами навченої моделі та надаються користувачеві).

3) Рівень довіри – дійсне число в діапазоні від нуля до одиниці.

6 Вихідні дані

Результати обчислень виводяться в інтерфейс системи у вигляді чітко позначених числових значень:

– Прогнозований розмір (LOC) – очікувана кількість рядків коду.

– Довірчий інтервал – нижня та верхня межа.

– Інтервал прогнозування – нижня та верхня межа.

Усі значення подаються у зрозумілому текстовому або табличному формат