МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2024 р.

КВАЛІФІКАЦІЙНАРОБОТА
на здобуття ступеня вищої освіти «магістр»

на тему: Нелінійна регресійна модель для оцінювання розміру модулів для електронної комерції на мові PHP та розробка програми для її реалізації.

Виконав: студент 6151м групи
_____ Тубольцев Р. О.
(підпис, ПІБ)

Керівник роботи:
_____ к.т.н., доцент
(посада, науковий ступень вчене звання)
_____ Пухалевич А. В.
(підпис, ПІБ)

Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«14» жовтня 2024 р.

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

Студенту Тубольцеву Роману Олександровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Нелінійна регресійна модель для оцінювання розміру модулів для електронної комерції на мові PHP та розробка програми для її реалізації».

Керівник роботи Пухалевич А. В.

Затверджені наказом ректора № 1070 УЧ уч від «11» _____10_____ 2024 р.

2. Термін подання роботи: 09.12.2024 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; Проект програмного забезпечення; 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 програмного забезпечення)

5. Перелік презентаційних матеріалів 1.Тема 2. Мета і 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 для оцінювання розмір веб-застосунків. 10. Обґрунтування необхідності проведення досліджень. 11. Емпіричні дані проектів e-commerce модулів, 12. Нормалізація даних та вилучення викидів, 13. Нелінійна регресійна модель, 14. Лінійна регресійна моделі, 15. Порівняння лінійної та нелінійної моделей, 16. Порівняння моделей для різних типів проектів, 17. Діаграма прецедентів застосунку для оцінювання e-commerce модулів, 18. Діаграма діяльності застосунку для оцінювання e-commerce модулів, 19. Діаграма класів застосунку для оцінювання e-commerce модулів. 20. Діаграма послідовності для варіанту використання «Виконати оцінювання розміру

e-commerce модуля», 21. Інтерфейс користувача застосунку для оцінювання e-commerce модулів, 22. Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 14.10.2024 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	16.10.2024	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	18.10.2024	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	21.10.2024	НС*
4.	Підготовка розділу з проекту програмного забезпечення	27.11.2024	
5.	Підготовка організаційно-економічного розділу	29.11.2024	
6.	Підготовка розділу з охорони праці	02.12.2024	
7.	Підготовка розділу з охорони навколишнього середовища	04.12.2024	
8.	Підготовка розділу Висновки	05.12.2024	
9.	Оформлення списку використаних джерел та додатків	06.12.2024	
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.)	09.12.2024	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	13.12.2024	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	16.12.2024	
13.	Подання на кафедру ПЗАС електронних версії наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	29.12.2024	

* - за результатами наукового стажування (НС), яке було з 02.09.2024 по 13.10.2024 р.

Студент _____ Тубольцев Р. О.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А. В.
(підпис) (ПІБ)

РЕФЕРАТ

Тубольцев Роман Олександрович

«Нелінійна регресійна модель для оцінювання розміру модулів для електронної комерції на мові PHP та розробка програми для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2024 р.

Обсяг роботи: 86 стор., 15 табл., 8 рис., 28 використаних джерел, 5 додатків

Актуальність теми: полягає в необхідності більш достовірного оцінювання розміру модулів для електронної комерції на мові PHP через відсутність побудованих моделей для такого виду проектів програмного забезпечення.

Мета дослідження: метою роботи полягає у підвищенні достовірності оцінки розміру модулів для електронної комерції, реалізованих мовою PHP, шляхом застосування нелінійної регресійної моделі.

Об’єкт дослідження: процес оцінювання розміру модулів для електронної комерції на мові PHP.

Предмет дослідження: нелінійні регресійні моделі для оцінювання розміру модулів для електронної комерції на мові PHP.

Методи дослідження: ймовірнісний аналіз, чисельні методи, регресійний аналіз, методи оптимізації, методи математичної статистики та об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: було удосконалено трьохфакторну нелінійну регресійну модель для оцінювання розміру модулів електронної комерції на PHP шляхом використання десяткового логарифмічного перетворення та вилучення викидів. У порівнянні з моделями для фреймворків Symfony, CakePHP та Yii, нова модель має вищий рівень точності, демонструючи більший коефіцієнт детермінації R^2, меншу похибку MMRE і кращий відсоток прогнозів PRED(0,25).

Практичне значення одержаних результатів. Результатом роботи стало створення програми, що забезпечує побудову нелінійних регресійних моделей для оцінювання розміру PHP-орієнтованих модулів електронної комерції. Цей інструмент дозволяє оцінювати розмір таких модулів уже на ранніх стадіях розробки.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи опубліковані на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи – 2024» (м. Миколаїв, 30-31 жовтня 2024 року).

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

Ключові слова: раннє оцінювання розміру, розмір програмного забезпечення, LOC, нормалізуючі перетворення, нелінійні регресійні моделі, e-commerce, PHP.

ABSTRACT

Tuboltsev Roman Oleksandrovych

«A nonlinear regression model for estimating the size of PHP-based e-commerce modules and development the software for its implementation »

Qualification work for obtaining a master's degree in specialty 121 – "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolaiv, 2024.

Volume: 86 p., 15 tables, 8 figures, 28 references, 5 appendices.

Topic Relevance: is the issue of more precise estimation of the size of PHP-based e-commerce modules due to the lack of built regression models for such project type.

Research goal. The aim of the work is to enhance the accuracy of size estimation for e-commerce modules implemented in PHP by applying a nonlinear regression model.

Object of research: the process of estimating the size of PHP-based e-commerce modules

Subject of research: the nonlinear regression models for estimating the size of PHP-based e-commerce modules.

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

The scientific novelty of the obtained results lies in the improvement of a three-factor nonlinear regression model for estimating the size of PHP-based e-commerce modules through the use of decimal logarithmic transformation and outlier removal. Compared to models for the Symfony, CakePHP, and Yii frameworks, the new model exhibits a higher level of accuracy, demonstrating a greater coefficient of determination (R^2), lower MMRE error, and a better prediction percentage PRED(0.25).

Practical value of obtained results. The result of the work was the development of a program that facilitates the construction of nonlinear regression models for estimating the size of PHP-oriented e-commerce modules. This tool enables the size estimation of such modules at the early stages of development.

Approbation of the thesis results. The 5rd All-Ukrainian scientific-practical Internet conference "Information technologies: models, algorithms, systems (ITMAS 2024)" (Mykolaiv, October 30-31, 2024).

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

Keywords: early LOC estimation, number of lines of code, LOC, normalizing transformations, non-linear regression models, e-commerce, PHP.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПК – Персональний комп’ютер

COCOMO – COnstructive COst Model

NOC – Number Of Classes

DIT – Depth of Inheritance Tree

KLOC – Thousand lines of code

LB – Lower Bound

UP – Upper Bound

LOC – Lines Of Code

MMRE – Mean Magnitude of Relative Error

MRE – Magnitude of Relative Error

PHP – Hypertext Preprocessor

PRED – Percentage of prediction

SMD – the Square Mahalanobis Distance

VIF – Variance Influence Factor

ВСТУП

Актуальність теми. Визначення розміру програмного забезпечення є важливим елементом в процесі управління проектами [1]. Розмір безпосередньо пов’язаний з обсягами витрат, витраченими зусиллями та використаними ресурсами. На основі розміру можна прогнозувати зусилля розробки за допомогою моделей COCOMO II [2] та ISBSG.

Існує багато способів оцінювання розміру: від підрахунку коду до аналізу функціональних характеристик. Проте часто їх застосування обмежується через брак даних або кваліфікованих спеціалістів. На сьогодні найбільш розповсюдженим методом є обчислення кількості рядків коду, але цей підхід не є універсальним. Методи на основі функціональних точок, зокрема FPA, пропонують альтернативні варіанти, вимірюючи функціональність у кількісному вигляді. Відсутність єдиного стандарту створює складнощі в прогнозуванні параметрів розробки [3-5].

Розробка e-commerce пакетів відіграє ключову роль у сучасній веб-індустрії, адже створення інтернет-магазину – це багатогранний процес, що охоплює управління товарами, оплату, обробку замовлень та інтеграцію зі сторонніми сервісами. Використання готових платформ та пакетів, таких як WooCommerce, Magento, Shopware чи більш компактних рішень, допомагає скоротити час розробки, зменшити витрати та уникнути рутинної роботи, дозволяючи зосередитися на створенні унікальної фукнціональності. Модульна структура та розширюваність цих рішень дають змогу адаптувати їх під специфічні бізнес-вимоги, а готові API значно полегшують інтеграцію з зовнішніми системам

PHP, будучи найбільш популярною мовою для веб-розробки, є природним вибором для створення e-commerce проектів [6]. Переваги включають доступність, широкий набір фреймворків та бібліотек, а також можливість створення динамічних веб-додатків із високою продуктивністю. Крім того, PHP має велику спільноту, яка забезпечує підтримку і швидкий доступ до готових

рішень. Постійне вдосконалення мови робить її затребуваним інструментом як для початківців, так і для досвідчених розробників.

Розмір програмного забезпечення, визначений за підходом LOC, став основою для створення регресійних моделей для різних мов програмування, зокрема PHP і її фреймворків (CakePHP, Yii, Laravel) [7-17]. Однак дослідження показали [16], що наявні моделі часто не дають точних результатів, що підкреслює важливість адаптації до конкретних фреймворків чи типу проектів.

Обраним нормалізуючим перетворенням для побудови нелінійної регресійної моделі є log10. Його ефективність доведена у моделях COCOMO, ISBSG, COCOMO II, а також у ряді інших підходів, зокрема для регресійних моделей на основі мови PHP. Використання log10 дозволяє зменшити вплив аномалій у вихідних даних і забезпечити більш достовірне оцінювання. Крім того, це перетворення забезпечує сумісність із попередніми дослідженнями, що дозволяє розширити та порівняти результати.

Мета та завдання дослідження: метою роботи полягає у підвищенні достовірності оцінки розміру модулів для електронної комерції, реалізованих мовою PHP, шляхом застосування нелінійної регресійної моделі.

Для досягнення цієї мети потрібно виконати такі завдання:

– Провести аналіз існуючих методів оцінювання розміру програмних застосунків, створених на PHP.

– Зібрати дані, необхідні для побудови нелінійної регресійної моделі для оцінювання розміру модулів для електронної комерції.

– Покращити нелінійну регресійну модель із використанням нормалізуючих перетворень.

– Розробити програмне забезпечення для реалізації розрахунків за створеною моделлю.

Об’єкт дослідження: процес оцінювання розміру модулів електронної комерції на мові PHP.

Предмет дослідження: нелінійні регресійні моделі для оцінювання розміру модулів для електронної комерції на мові PHP.

Методи дослідження: ймовірнісний аналіз, чисельні методи, регресійний аналіз, методи оптимізації, методи математичної статистики та об'єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: було удосконалено трьохфакторну нелінійну регресійну модель для оцінювання розміру модулів електронної комерції на PHP шляхом використання десяткового логарифмічного перетворення та вилучення викидів. У порівнянні з моделями для фреймворків Symfony, CakePHP та Yii, нова модель має вищий рівень точності, демонструючи більший коефіцієнт детермінації R^2, меншу похибку MMRE і кращий відсоток прогнозів PRED(0,25).

Практичне значення одержаних результатів. Результатом роботи стало створення програми, що забезпечує побудову нелінійних регресійних моделей для оцінювання розміру PHP-орієнтованих модулів електронної комерції. Цей інструмент дозволяє оцінювати розмір таких модулів уже на ранніх стадіях розробки.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною науковою працею. Усі наукові результати отримано автором особисто. У праці [18], яка була опублікована у співавторстві, автору належить нелінійна регресійна модель для оцінювання розміру модулів для електронної комерції на мові PHP на основі нормалізуючого перетворення десяткового логарифму.

Апробація результатів роботи. Основні положення й результати кваліфікаційної роботи опубліковані на V Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи – 2024» (м. Миколаїв, 30-31 жовтня 2024 року).

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

1 ОЦІНКА ІСНУЮЧИХ МОДЕЛЕЙ ДЛЯ ВИЗНАЧЕННЯ РОЗМІРУ ПРОГРАМНИХ ЗАСТОСУНКІВ І АРГУМЕНТАЦІЯ ПОТРЕБИ В ПРОВЕДЕННІ ДОСЛІДЖЕНЬ

1.1 Огляд існуючих методів і математичних моделей для оцінювання розміру програмних застосунків

Визначення розміру програмного забезпечення є важливим є важливим елементом в процесі управління проектами [1]. Розмір безпосередньо пов’язаний з обсягами витрат, витраченими зусиллями та використаними ресурсами. На основі розміру можна прогнозувати зусилля розробки за допомогою моделей COCOMO II та ISBSG [2].

Існує багато способів оцінювання розміру: від підрахунку коду до аналізу функціональних характеристик. Проте часто їх застосування обмежується через брак даних або кваліфікованих спеціалістів. На сьогодні найбільш розповсюдженим методом є обчислення кількості рядків коду, але цей підхід не є універсальним. Методи на основі функціональних точок, зокрема FPA, пропонують альтернативні варіанти, вимірюючи функціональність у кількісному вигляді. Відсутність єдиного стандарту створює складнощі в прогнозуванні параметрів розробки [3 - 5].

Розробка інтернет-магазину – складний та трудомісткий процес, який вимагає від розробника продумати безліч аспектів: від управління каталогами товарів до інтеграції з платіжними системами та зовнішніми сервісами. Замість того щоб створювати весь функціонал з нуля, розробники можуть використовувати готові пакети та платформи, які значно спрощують процес і дозволяють зосередитися на бізнес-логіці та інтерфейсі користувача. Розглянемо з прикладами, в чому саме використання готових пакетів для електронної комерції може допомогти розробнику.

Економія часу та ресурсів

Однією з ключових переваг використання готових e-commerce платформ та пакетів, таких як Magento, WooCommerce або Bagisto є значне скорочення часу на розробку. Замість того, щоб проектувати базові функції – кошик, управління замовленнями, каталог товарів – ви отримуєте їх «з коробки». Це особливо важливо на старті, коли важливо запустити MVP (мінімально життєздатний продукт) за короткий термін.

Наприклад, WooCommerce-плагін для WordPress – дозволяє налаштувати магазин із базовим функціоналом буквально за кілька годин. Це ідеальне рішення для невеликих магазинів із обмеженим бюджетом. Більш складні платформи, такі як Sylius або Magento, вимагають більше часу на впровадження, але натомість надають величезні можливості для кастомізації та масштабування.

Розширюваність та гнучкість кастомізації

Багато сучасних e-commerce рішень, такі як Bagisto, Sylius і Shopware, побудовані за модульним принципом. Це означає, що можна легко додавати нові модулі або змінювати існуючі відповідно до вимог проекту. Такий підхід дозволяє гнучко адаптувати платформу під унікальні бізнес-потреби.

Наприклад, Sylius, заснований на Symfony, дозволяє розробникам створювати кастомні рішення з нуля, використовуючи вбудовані компоненти. Shopware та Aimeos підтримують headless-архітектуру, що дає можливість розробляти складні інтерфейси та інтеграції, зберігаючи при цьому високу продуктивність.

Масштабованість

Для інтернет-магазинів, які планують довгострокове зростання, важливим критерієм вибору платформи є масштабування. Рішення як Magento і Shopware здатні справлятися з високим трафіком і великою кількістю даних, при цьому надаючи інструменти для ефективного управління каталогом товарів, що росте.

Magento, наприклад, використовується багатьма великими брендами завдяки своїй потужній архітектурі. Однак така масштабованість вимагає

суттєвих ресурсів та високої кваліфікації розробників, що варто враховувати під час виборів платформи.

Простота інтеграції із зовнішніми сервісами

Сучасні e-commerce платформи надають API, які полегшують інтеграцію з іншими системами: CRM, WMS, платіжними шлюзами та аналітичними сервісами. Це дозволяє легко впроваджувати додаткові інструменти для керування магазином та аналізувати ключові показники. Aimeos і Shopware, наприклад, мають добре документовані API, які дозволяють інтегрувати платформу з мобільними програмами, платіжними системами або сторонніми маркетинговими сервісами. Це робить їх придатними для проектів, де потрібний високий рівень автоматизації.

Доступ до екосистеми та спільноти

Використання популярних рішень, таких як WooCommerce, Magento або PrestaShop, відкриває доступ до великої екосистеми плагінів, тем і готових рішень. Це дозволяє швидко додавати нові функції, такі як підтримка нових мов, валют або способів оплати без необхідності розробляти їх самостійно.

Крім того, активна спільнота розробників забезпечує підтримку, ділиться готовими рішеннями та допомагає вирішувати проблеми. Це особливо цінно для розробників-початківців, які можуть багато чому навчитися, працюючи з такими платформами.

PHP залишається найбільш популярною мовою для веб-розробки, особливо в галузі e-commerce, завдяки своїй доступності, простоті та спеціалізованій спрямованості на створення серверних рішень [6]. Сильні сторони PHP – це низький поріг входу для початківців, велика екосистема бібліотек та фреймворків (таких як Laravel, Symfony), а також інтеграція з популярними веб-серверами (Apache, Nginx) та СУБД (MySQL, PostgreSQL). PHP відмінно справляється з динамічним рендерингом сторінок, побудовою RESTful API, створенням застосунків, що масштабуються, і підтримує безліч готових рішень для платіжних систем, кошиків та управління замовленнями. Широка поширеність мови PHP у веб, активна спільнота та постійний розвиток мови (наприклад, додавання суворої

типізації та покращення продуктивності в останніх версіях) роблять її ідеальним вибором для розробки e-commerce сайтів, сервісів та пакетів.

Визначення розміру програмного забезпечення за підходом LOC стало предметом численних досліджень, у межах яких було створено лінійні та нелінійні регресійні моделі для таких мов програмування, як C++, Java, Visual Basic [7 - 9]. У більшості випадків ці моделі базувалися на метриках діаграм класів, що дозволяє ще на етапі проектування аналізувати структуру класів. Для PHP і її фреймворків, таких як CakePHP, Yii, CodeIgniter та Laravel, також розроблено значну кількість моделей [10 - 17]. Однак, згідно з результатами дослідження [16], хоча й була висунута гіпотеза про їх ефективність, спроби застосування цих моделей не призвели до достовірних результатів, що ставить питання про необхідність побудови таких моделей для фремворку чи типу проектів.

Обраним нормалізуючим перетворенням для побудови нелінійної регресійної моделі є log10. Його ефективність доведена у моделях COCOMO, ISBSG, COCOMO II, а також у ряді інших підходів, зокрема для регресійних моделей на основі мови PHP. Використання log10 дозволяє зменшити вплив аномалій у вихідних даних і забезпечити більш достовірне оцінювання. Крім того, це перетворення забезпечує сумісність із попередніми дослідженнями, що дозволяє розширити та порівняти результати.

Для нормалізованих даних можна застосувати лінійну регресійну модель [19]:

$$Y = \hat{Y} + \varepsilon = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_k X_k + \varepsilon, \quad (1.1)$$

де $\hat{Y}$ – результат передбачення за рівнянням лінійної регресії,

b_i $(i = \overline{0,k})$ – параметри лінійного рівняння регерсії,

X_i $(i = \overline{1,k})$ – значення незалежної змінної,

ε – гаусівська випадкова величина,

k – кількість предикторів у моделі.

Довірчий інтервал лінійної регресії для даних, що є нормальними, має вигляд [19]:

$$\hat{Y}_i \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{ \frac{1}{N} + (\mathbf{x}^+)^T [\mathbf{S_{XX}}]^{-1} (\mathbf{x}^+) \right\}^{1/2}, \tag{1.2}$$

а інтервал передбачення [19]:

$$\hat{Y}_i \pm t_{\alpha/2,\mathrm{v}} S_Y \left\{ 1 + \frac{1}{N} + (\mathbf{x}^+)^T [\mathbf{S_{XX}}]^{-1} (\mathbf{x}^+) \right\}^{1/2}, \tag{1.3}$$

де $S_Y^2 = \frac{1}{\mathrm{v}} \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2$;

$\mathrm{v} = N - k - 1$;

$\mathbf{S_{XX}} - k \times k$ матриця [19]:

$$\mathbf{S_{XX}} = \begin{pmatrix} S_{X_1X_1} & S_{X_1X_2} & \dots & S_{X_1X_k} \\ S_{X_1X_2} & S_{X_2X_2} & \dots & S_{X_2X_k} \\ \dots & \dots & \dots & \dots \\ S_{X_1X_k} & S_{X_2X_k} & \dots & S_{X_kX_k} \end{pmatrix}, \tag{1.4}$$

де $S_{X_qX_r} = \sum_{i=1}^{N} \left[X_{q_i} - \bar{X}_q \right] \left[X_{r_i} - \bar{X}_r \right]$,

$q, r = \overline{1, k}$.

У разі, коли початкові дані не мають гаусівського розподілу, виникає потреба у додаткових зусиллях для побудови нелінійної регресії. Для цього можна використати три основні методи: метод підбору, лінеаризуючі або нормалізуючі перетворення. Серед них нормалізуючі перетворення є найбільш ефективними, оскільки зазвичай дозволяють досягти прийнятного рівня достовірності моделі без значних часових затрат.

Алгоритм методів на основі нормалізуючих перетворень складається з кількох кроків. На першому етапі дані перетворюються задля забезпечення їх нормалізації. Потім видаляються викиди, а очищений набір даних використовується для побудови лінійної регресії. Залишки моделі перевіряються на відповідність нормальному розподілу за допомогою критерію Пірсона. Якщо відповідність не досягається, проблемний рядок даних (найбільших за модулем) виключається, і процес повторюється. Після завершення цих кроків застосовується зворотне перетворення для створення фінальної нелінійної моделі. На останньому етапі будується інтервал прогнозування, і при необхідності виключаються дані, що виходять за його межі [19, 20].

Перетворення використовується для нормалізації даних [19]:

$$\mathrm{T} = \psi(\mathrm{P}), \tag{1.5}$$

де $\mathrm{P} = \{Y, X_1, X_2, \dots, X_k\}^T$ – вектор ненормалізованих даних,

$\mathrm{T} = \{Z_Y, Z_1, Z_2, \dots, Z_k\}^T$ – вектор нормалізованих даних,

$\psi = \{\psi_Y, \psi_1, \psi_2, \dots, \psi_k\}^T$ – застосоване нормалізуюче перетворення.

Зворотнім до (1.5) є перетворення [19]:

$$\mathrm{P} = \psi^{-1}(\mathrm{T}). \tag{1.6}$$

Готова нелінійна регресійна модель, що будувалась з використанням нормалізуючих перетворень має вигляд [25]:

$$Y = \psi_Y^{-1}(\hat{Z}_Y + \varepsilon) = \psi_Y^{-1}(b_0 + b_1 Z_1 + b_2 Z_2 + \dots + b_k Z_k + \varepsilon), \tag{1.7}$$

де ψ_Y – перша компонента вектору ψ перетворення (1.5).

Довірчий інтервал нелінійної регресії має вигляд [19]:

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{\frac{1}{N} + (\boldsymbol{z}_X^+)^T[\mathbf{S_{ZZ}}]^{-1}(\boldsymbol{z}_X^+)\right\}^{1\backslash 2}\right). \quad (1.8)$$

Інтервал передбачення нелінійної регресії має вигляд [19]:

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\frac{\alpha}{2},\nu} S_{Z_Y}\left\{1 + \frac{1}{N} + (\boldsymbol{z}_X^+)^T[\mathbf{S_{ZZ}}]^{-1}(\boldsymbol{z}_X^+)\right\}^{1\backslash 2}\right). \quad (1.9)$$

Таким чином, розглянувши існуючий теоретичний матеріал з теми дослідження, можна переходити до обгрунтування необхідності проведення власних досліджень.

1.2 Обґрунтування необхідності проведення досліджень

Практичне застосування лінійного регресійного аналізу для визначення розміру програмного забезпечення часто не відповідає ключовим умовам, наприклад, нормальності розподілу залишків регресії, яка рідко зустрічається в реальних даних. У таких випадках виникає необхідність переходу до нелінійних моделей. Крім того, реальні дані в розробці програмного забезпечення часто містять викиди або мають негаусівський розподіл. Це додатково ускладнює застосування лінійної регресії та вимагає переходу до більш гнучких підходів, таких як нелінійні регресійні моделі [20, 21].

PHP широко використовується у веб-розробці [6], і для неї існує значна кількість нелінійних регресійних моделей, які базуються на метриках діаграм класів [12 - 17]. Як доведено в роботі [16], розробка регресійної моделі для окремого фреймворку має сенс, адже універсальної моделі не існує, й навіть моделі для фреймворків в межах однієї мови програмування погано враховують специфіку, що також справедливо й для типу проектів. Це підтверджує необхідність детального дослідження цієї теми.

1.3 Висновки до розділу 1

У розділі 1 проведено аналіз важливості оцінювання розміру програмного забезпечення у процесі його розробки, а також коротко розглянуто сучасні методи оцінювання цього показника. Розглянуто роль PHP у створенні модулів для електронної комерції, розглянуто питання використання регресійного аналізу для оцінювання кількості рядків коду (LOC). Обґрунтовано важливість використання нормалізуючих перетворень для підвищення достовірності прогнозування, що обумовлено складністю застосування лінійних регресійних моделей в реальних умовах. Отримані висновки вказують на необхідність розробки моделей, які враховують специфіку окремих фреймворків, типів проектів, що є актуальним завданням для подальших досліджень у цій галузі. Такий підхід сприяє вдосконаленню процесів оцінювання розміру програмного забезпечення, допомагає підвищити ефективність планування ресурсів про роботі над проектом програмного забезпечення.

2 РЕАЛІЗАЦІЯ НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ РОЗМІРУ МОДУЛІВ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА PHP

2.1 Дослідження даних на наявність мультиколінеарності та аномалій для побудови нелінійної регресійної моделі оцінювання PHP-модулів електронної комерції

Нелінійну регресійну модель для аналізу модулів для електронної комерції мові на PHP буде побудовано на основі 40 проєктів, вибраних із платформи GitHub (https://github.com/). Використання програмного забезпечення PhpMetrics дозволило отримати ключові метрики діаграми класів: кількість рядків коду у тисячах (KLOC, Y), кількість класів (X_1), середню кількість методів на клас (X_2), а також глибину дерева успадкування (DIT, X_3). Ці дані ляжуть в основу створення моделі. Дані наведено у таблиці 2.1

Таблиця 2.1 – Отримані метрики модулів для електронної комерції

№	KLOC (Y)	Classes (X_1)	Methods by Class (X_2)	DIT (X_3)	SMD_X
1	2	3	4	5	6
1	0,675	14	6,21	1,2	9,4090
2	1,173	44	1,41	1,95	4,0649
3	1,179	49	1,67	2	4,4268
4	2,359	40	2,93	1,46	2,4741
5	2,636	57	2,44	1,91	2,6088
6	3,013	118	1,69	1,68	2,8137
7	3,045	85	3,14	1,67	1,6209
8	3,365	45	3,64	1,13	5,8592
9	3,598	65	3,45	1,85	2,1699
10	5,157	108	4,06	1,57	0,8214
11	5,86	121	3,49	1,68	0,9938
12	8,304	234	2,37	1,63	1,4597
13	9,571	148	3,49	1,81	1,7751
14	12,23	211	3,53	2,18	5,7111

Продовження табл. 2.1

1	2	3	4	5	6
15	16,03	274	4,01	1,48	0,1483
16	21,244	451	3,34	1,22	1,2677
17	23,508	245	4,02	1,23	0,9832
18	23,886	407	3,85	1,14	1,5372
19	25,147	764	3,2	1,21	2,8418
20	27,528	185	5,14	1,14	3,1287
21	34,612	186	7,53	1,17	3,6633
22	38,519	263	6,02	1,23	1,8018
23	40,734	1012	2,74	1,25	2,0359
24	45,691	603	4,93	1,59	1,1274
25	49,668	1046	3089	1,25	**37,4408**
26	51,83	390	6,73	2,08	8,1791
27	52,553	902	4,07	1,15	1,2962
28	69,18	1913	3,43	1,19	3,4212
29	69,581	718	4,85	1,25	0,3808
30	78,161	240	11,99	1,43	7,0704
31	102,746	2347	5,1	1,11	3,2908
32	110,622	850	4,22	1,32	1,1529
33	150,568	1428	4,26	1,06	1,8514
34	176,331	441	13,29	1,26	7,6108
35	255,584	1430	7,77	1,16	2,1908
36	350,442	4743	3,24	1,1	3,3813
37	481,45	4297	5,01	1,35	3,4387
38	491,849	3113	9,67	1,02	2,9528
39	927,548	9353	3,84	1,01	4,9268
40	968,834	11416	4,1	1,37	6,6716

Для обчислення квадрату відстані Махаланобіса (SMD) для і-тої точки багатовимірного набору даних застосовується формула [19]:

$$d_i^2 = (X_i - \overline{X})^T S_N^{-1} (X_i - \overline{X}) , \qquad (2.1)$$

де X_i – i- та точка вибірки,

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

S_N – коваріаційна матриця,

N – розмір вибірки, вектор чотирьох компонентний.

Вибіркова коваріаційна матриця обчислюється як [19]:

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

Для ряду 25 значення d_i^2 перевищують критичний поріг (9,488) для рівня значущості 0,005, що вказує на ненормальний розподіл метрик.

Для підтвердження цього висновку розраховано багатовимірний ексцес $\hat{\beta}_2$ за формулою [22]:

$$\hat{\beta}_2 = \frac{1}{N}\sum_{i=1}^{N}\{(X_i - \bar{X})^T S_N^{-1}(X_i - \bar{X})\}^2. \quad (2.3)$$

Розрахунок дав значення 64,446, яке перевищує критичний рівень 24 ($\beta_2 = m(m+2), m = 4$). Таким чином, дані з метрик модулів електронної комерції на мові PHP підтверджують наявність негаусівського розподілу.

Мультиколінеарність є проблемою в регресійному аналізі, що виникає при існуванні сильного лінійного зв’язку між факторами. Це ускладнює визначення впливу кожного фактора на залежну змінну.

Для перевірки на мультиколінеарність використовуються коефіцієнти Variance Inflation Factors (VIFs). Вони є елементами головної діагоналі оберненої коваріаційної матриці для *k*-предикторів у моделі. Значення VIFs до 5 вважаються безпечними, тоді як значення понад 10 вказують на наявність проблеми [23].

У моделі, що включає предиктори X_1, X_2, X_3 обчислені значення VIFs становлять 1,407, 1,268 і 1,417 відповідно Оскільки всі значення знаходяться в межах допустимого діапазону, мультиколінеарність у моделі відсутня.

Обчислення SMD для ненормалізованих даних вказує на відхилення розподілу від нормального. Для обробки ненормальних даних застосовувався метод, який поєднує нормалізуючі перетворення та розрахунок SMD [24, 25].

Формула для нормалізованих даних [19]:

$$d_i^2 = (Z_i - \overline{Z})^T S_N^{-1} (Z_i - \overline{Z}) , \quad (2.4)$$

де Z_i – i-та точка нормалізованого вектора Z,

$\overline{Z}$ – вектор вибіркових середніх,

S_N – вибіркова коваріаційна матриця, вектор є чотирьох-компонентним.

Формула для S_N [19]:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z_i - \overline{Z}) (Z_i - \overline{Z})^T. \quad (2.5)$$

Вектор Z формується за допомогою логарифмічного перетворення log10, що визначається як $\psi = \{\lg Y, \lg X_1, \lg X_2, \lg X_3\}^T$.

Для багатовимірного нормального розподілу значення SMD відповідають χ^2-розподілу. У великих вибірках ($N - m > 25$) величини d_i^2 наближаються до значень випадкової величини $\chi^2_{m,\alpha}$, де $\chi^2_{m,\alpha}$ є квантилем χ^2-розподілу для заданого рівня значущості α [26].

На основі аналізу даних усі викиди видалені за три ітерації, пошук викидів наведено в таблиці 2.2.

Таблиця 2.2 – Видалення викидів

№	SMD_1	SMD_2	SMD_3	№	SMD_1	SMD_2	SMD_3
1	2	3	4	5	6	7	8
1	9,409	15,390	-	21	3,663	3,565	3,934
2	4,065	6,893	6,948	22	1,802	1,876	1,906
3	4,427	4,715	4,569	23	2,036	2,009	1,932
4	2,474	3,440	3,592	24	1,127	1,938	1,876
5	2,609	3,151	3,070	25	37,441	-	-
6	2,814	3,554	3,453	26	8,179	9,148	9,026
7	1,621	2,313	3,113	27	1,296	1,404	1,608
8	5,859	6,761	7,705	28	3,421	4,416	4,546

Продовження табл. 2.2

1	2	3	4	5	6	7	8
9	2,170	2,138	2,202	29	0,381	0,358	0,326
10	0,821	2,135	3,276	30	7,070	7,339	7,311
11	0,994	1,254	1,502	31	3,291	8,078	9,282
12	1,460	1,485	1,427	32	1,153	3,173	3,923
13	1,775	1,719	1,666	33	1,851	2,925	2,960
14	5,711	5,798	5,789	34	7,611	7,616	7,507
15	0,148	0,433	0,632	35	2,191	2,185	2,105
16	1,268	1,228	1,480	36	3,381	4,634	5,467
17	0,983	1,979	1,908	37	3,439	3,340	4,151
18	1,537	1,478	1,909	38	2,953	5,015	5,335
19	2,842	3,640	4,226	39	4,927	6,392	7,580
20	3,129	4,586	4,483	40	6,672	6,500	8,276

Подальший аналіз виконуватиметься на основі очищеного набору даних із 38 проектів модулів електронної комерції на мові PHP, з якого вилучено викиди. Для перевірки можливості застосування лінійної регресії необхідно оцінити нормальність розподілу залишків регресії ε. У разі великої вибірки ($n > 30$) нульову гіпотезу щодо нормальності можна перевірити за допомогою критерію Пірсона [19].

Величина χ^2 визначається як [19]:

$$\chi^2 = \sum_{j=1}^{m} \frac{\left(n_j - np_j\right)^2}{np_j}, \tag{2.6}$$

де m – кількість інтервалів у діапазоні$[x_{min}, x_{max}]$,

n_j – частота попадання у j-ий інтервал,

p_j – ймовірність попадання в цей інтервал.

Зазвичай кількість інтервалів обчислюється однією з двох формул [19]:

– Старджеса: $m = \log_2 n + 1 = 3{,}3lgn + 1$,

– Брукса і Карузера: $m = 5lgn$.

Ймовірності p_j обчислюються як [19]:

$$\boldsymbol{p_j = \int_{x_{j-1}}^{x_j} f(x)dx,} \tag{2.7}$$

де $f(x)$ – функція щільності ймовірності.

Ймовірність p_j обчислюється інтегруванням функції щільності [19]:

$$\boldsymbol{f(x) = \frac{1}{\sigma_x\sqrt{2\pi}} * e^{\frac{-(x-m_x)^2}{2\sigma_x^2}},} \tag{2.8}$$

де m_x – математичне сподівання випадкової величини,

σ_x – середьоквадратичне відхилення випадкової величини.

Ступені свободи для критерію обчислюються як $\nu = m - k - 1$, де $k = 2$ для нормального розподілу.

Остаточне значення χ^2, що спостерігається, порівнюється з критичним для відповідного рівня значущості α. Якщо $\chi^2 < \chi^2_{\text{кр}}$, приймається нульова гіпотеза про нормальність розподілу, інакше – відкидається.

2.2 Розробка нелінійної регресійної моделі для оцінки розміру модулів електронної комерції на PHP

Для побудови нелінійної регресійної моделі необхідно побудувати лінійну регресійну модель для нормалізованих даних без викидів. Модель можна описати наступним рівнянням [19]:

$$Z_Y = \hat{Z}_Y + \varepsilon = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \hat{b}_3 Z_3 + \varepsilon, \tag{2.9}$$

де $Z_y = \lg Y; Z_i = \lg X_i, i = \overline{1,3}$.

Для побудованої моделі слід сформулювати та перевірити нульову гіпотезу про нормальність розподілу залишків регресії ε. Емпіричне значення критерію Пірсона, розраховане за формулою (2.6) має бути меншим за критичне для 3 ступенів свободи та рівня значущості 0,05, тобто $\chi^2 = 7{,}814$. У випадку відхилення нульової гіпотези необхідно відкинути найбільше за модулем значення ε як викид.

За лінійним рівняння (2.9) та перетворенням (1.5) отримуємо [19]:

$$Y = 10^{\varepsilon + \hat{b}_0} X_1^{\hat{b}_1} X_2^{\hat{b}_2} X_3^{\hat{b}_3}. \quad (2.10)$$

Для побудованої нелінійної регресійної моделі слід знайти інтервали передбачення. Всі значення, що виходять за межі інтервалів слід відкинути як видики й розпочати з етапу пошуку викидів з використанням SMD.

Розпочнемо ітераційний процес побудови нелінійної регресійної моделі.

На першій ітерації для даних без викидів побудовано лінійну регресійну модель для нормалізованих даних, її оцінки параметрів: $\hat{b}_0 = -1{,}8433, \hat{b}_1 = 0{,}9896, \hat{b}_2 = 1{,}2648, \hat{b}_3 = -0{,}1900$. Емпіричне значення $\chi^2 = 0{,}81$, нульова гіпотеза прийнята. Побудовані інтервали передбачення показали один викид – проект 31, відкинемо його й почнемо з першого кроку.

На другій ітерації для емпіричних даних (без проектів 25, 1, 31) викидів за SMD не виявлено. Значення величини $\chi^2 = 9{,}26$, проект 19 слід відкинути та почати з першого кроку.

Не третій ітерації для емпіричних даних (без проектів 25, 1, 31, 19) викидів за SMD не виявлено. Значення величини $\chi^2 = 7{,}21$, нульова гіпотеза прийнята. Інтервали передбачення показали один видик – проект 28, відкинемо його й почнемо з першого кроку.

На четвертій ітерації для емпіричних даних (без проектів 25, 1, 31, 19, 28) викидів за SMD не виявлено. Значення величини $\chi^2 = 4{,}88$, нульова гіпотеза прийнята. Інтервали передбачення не показали викидів, отже нелінійна регресійна модель побудована, її оцінки параметрів: $\hat{b}_0 = -1{,}8137, \hat{b}_1 = 1{,}0044, \hat{b}_2 = 1{,}2014, \hat{b}_3 = -0{,}2958$.

Модель оцінювалася за такими показниками:

– $R^2 = 0{,}943$;

– $MMRE = 0{,}188$;

– $PRED(0{,}25) = 0{,}714$.

Оцінки моделі R^2, $MMRE$ відповідають стандартним критеріям якості ($MMRE \leq 0{,}25$, $R^2 > 0{,}75$), але значення $PRED(0{,}25)$ дещо нижче за прийнятне ($PRED(0{,}25) > 0{,}75$) [27, 28].

Довірчий інтервал та інтервал передбачення були розраховані з використанням матриці S_{ZZ} за (1.8) і (1.9)

$$S_{ZZ} = \begin{pmatrix} 15{,}0478 & 1{,}8291 & -1{,}2619 \\ 1{,}8291 & 1{,}5974 & -0{,}3112 \\ -1{,}2619 & -0{,}3112 & 0{,}3126 \end{pmatrix}.$$

У рамках цього дослідження створено багатофакторну лінійну регресійну модель для визначення розміру модулів для електронної комерції на мові PHP. Модель базується на припущенні про нормальний розподіл метрик і має такий вигляд [19]:

$$Y = \hat{Y} + \varepsilon = \hat{b}_0 + \hat{b}_1 X_1 + \hat{b}_2 X_2 + \hat{b}_3 X_3 + \varepsilon. \quad (2.12)$$

Маємо наступні оцінки параметрів: $\hat{b}_0 = -48{,}027, \hat{b}_1 = 0{,}09154, \hat{b}_2 = 14{,}5208, \hat{b}_3 = -4{,}4004$.

Для перевірки можливості застосування лінійних регресійних моделей було сформульовано нульову гіпотезу про нормальність розподілу залишків регресії ε.

Емпіричне значення критерію Пірсона, розраховане за формулою (2.6) для ненормалізованих даних із 35 проектів, становить $\chi^2 = 108{,}3$, тоді як критичне значення для 3 ступенів свободи та рівня значущості 0,05 дорівнює $\chi^2 = 7{,}814$. Оскільки $\chi^2 > \chi^2_{\text{кр}}$, нульова гіпотеза не приймається. Для залишків регресії було визначено наступні характеристики: математичне сподівання 128,53 і середньоквадратичне відхилення 240,982.

Модель оцінювалася за такими показниками:

– $R^2 = 0{,}972$;

– $MMRE = 2{,}499$;

– $PRED(0{,}25) = 0{,}4$.

Значення $MMRE$ і $PRED(0{,}25)$ не відповідають прийнятним межам ($MMRE \leq 0{,}25$, $PRED(0{,}25) > 0{,}75$), що разом із ненормальністю розподілу залишків свідчить про низьку достовірність цієї моделі.

Довірчий інтервал та інтервал передбачення були розраховані з використанням матриці S_{XX}:

$$S_{XX} = \begin{pmatrix} 221208264{,}743 & 2958{,}120 & -9655{,}489 \\ 2958{,}120 & 239{,}072 & -9{,}962 \\ -9655{,}489 & -9{,}962 & 3{,}703 \end{pmatrix}.$$

У таблиці 2.3 наведено результати аналізу довірчих інтервалів багатофакторних лінійної та нелінійної регресійних моделей, розрахованих за формулами (1.2) і (1.8). Встановлено, що нелінійна регресія забезпечує менші довірчі інтервали у 85,71% випадків (30 з 35 рядків).

Таблиця 2.3 – Довірчі інтервали лінійної та нелінійної регресії

№	KLOC (Y)	Межі довірчих інтервалів				Ширина інтервалу	
		Лінійна регресія		Нелінійна регресія		Лінійна регресія	Нелінійна регресія
		LB	UB	LB	UB		
1	2	3	4	5	6	7	8
1	1,173	-61,215	-3,002	0,696	1,043	58,213	**0,348**
2	1,179	-58,225	2,034	0,959	1,390	60,260	**0,432**

Продовження таблиці 2.3

1	2	3	4	5	6	7	8
3	2,359	-27,728	11,235	1,725	2,390	38,964	**0,665**
4	2,636	-41,847	10,273	1,850	2,496	52,120	**0,646**
5	3,013	-43,169	3,000	2,522	3,524	46,169	**1,003**
6	3,045	-20,976	16,968	4,028	5,075	37,944	**1,047**
7	3,365	-22,839	30,785	2,540	4,032	53,623	**1,493**
8	3,598	-23,458	23,209	3,264	4,313	46,667	**1,050**
9	5,157	-2,651	30,450	7,158	8,886	33,101	**1,728**
10	5,86	-12,274	24,930	6,557	8,142	37,204	**1,585**
11	8,304	-19,637	20,883	7,929	10,173	40,520	**2,244**
12	9,571	-13,760	30,212	7,716	9,923	43,972	**2,207**
13	12,23	-23,969	49,855	9,771	14,709	73,824	**4,938**
14	16,03	12,659	44,855	18,726	22,157	32,196	**3,432**
15	21,244	13,216	59,517	25,398	32,118	46,300	**6,720**
16	23,508	5,801	48,897	17,175	21,657	43,096	**4,482**
17	23,886	15,327	64,870	27,352	35,540	49,543	**8,188**
18	27,528	15,287	61,751	17,137	23,307	46,465	**6,171**
19	34,612	48,283	98,083	26,569	37,457	49,800	**10,888**
20	38,519	37,782	78,293	29,663	38,150	40,511	**8,487**
21	40,734	55,822	101,876	43,616	58,091	46,054	**14,475**
22	45,691	55,187	88,280	49,850	63,904	33,093	**14,054**
23	51,83	36,368	116,086	38,112	62,769	79,718	**24,658**
24	52,553	65,930	111,146	65,670	83,240	45,216	**17,570**
25	69,581	64,266	100,913	64,422	77,836	36,646	**13,414**
26	78,161	95,984	187,501	53,604	84,128	91,517	**30,524**
27	110,622	68,248	102,173	63,529	76,784	33,926	**13,255**
28	150,568	115,047	164,589	109,972	146,470	49,542	**36,498**
29	176,331	128,551	230,964	116,215	181,793	102,413	**65,579**
30	255,584	166,533	214,525	221,403	293,130	47,993	71,728
31	350,442	399,064	457,194	248,533	366,208	58,130	117,675
32	481,45	388,864	434,986	364,560	516,903	46,122	152,343
33	491,849	338,689	406,744	626,663	901,918	68,055	275,255
34	927,548	809,323	908,713	603,882	932,726	99,390	328,844
35	968,834	986,804	1113,113	712,720	1152,694	126,309	439,974

Інтервали передбачення та їх ширини, представлені в таблиці 2.4 (розраховані за формулами (1.3) і (1.9)), також демонструють перевагу нелінійної регресії: у 85,71% випадків (30 з 35 рядків) ширина інтервалів менша.

Таблиця 2.4 – Інтервали передбачення лінійної та нелінійної регресії

№	KLOC (Y)	Межі інтервалів передбачення				Ширина інтервалу	
		Лінійна регресія		Нелінійна регресія		Лінійна регресія	Нелінійна регресія
		LB	UB	LB	UB		
1	2	3	4	5	6	7	8
1	1,173	-123,08	58,86	0,696	1,043	181,943	**0,934**
2	1,179	-119,40	63,21	0,959	1,390	182,608	**1,249**
3	2,359	-96,61	80,12	1,725	2,390	176,728	**2,161**
4	2,636	-105,83	74,26	1,850	2,496	180,086	**2,267**
5	3,013	-109,31	69,14	2,522	3,524	178,455	**3,183**
6	3,045	-90,26	86,25	4,028	5,075	176,506	**4,678**
7	3,365	-86,29	94,24	2,540	4,032	180,527	**3,594**
8	3,598	-89,42	89,17	3,264	4,313	178,584	**3,934**
9	5,157	-73,86	101,66	7,158	8,886	175,528	**8,222**
10	5,86	-81,85	94,50	6,557	8,142	176,348	**7,533**
11	8,304	-87,92	89,16	7,929	10,173	177,077	**9,337**
12	9,571	-80,72	97,18	7,716	9,923	177,899	**9,102**
13	12,23	-80,82	106,70	9,771	14,709	187,522	**13,161**
14	16,03	-58,92	116,44	18,726	22,157	175,360	**20,784**
15	21,244	-52,88	125,61	25,398	32,118	178,489	**29,579**
16	23,508	-61,49	116,19	17,175	21,657	177,684	**19,959**
17	23,886	-49,58	129,78	27,352	35,540	179,357	**32,527**
18	27,528	-50,75	127,78	17,137	23,307	178,531	**21,141**
19	34,612	-16,53	162,90	26,569	37,457	179,428	**33,781**
20	38,519	-30,50	146,58	29,663	38,150	177,075	**34,995**
21	40,734	-10,36	168,06	43,616	58,091	178,425	**52,896**
22	45,691	-16,03	159,50	49,850	63,904	175,527	**58,663**
23	51,83	-18,73	171,19	38,112	62,769	189,920	**55,839**
24	52,553	-0,57	177,64	65,670	83,240	178,210	**76,615**
25	69,581	-5,53	170,71	64,422	77,836	176,231	**72,557**
26	78,161	44,16	239,33	53,604	84,128	195,166	**75,027**
27	110,622	-2,63	173,05	63,529	76,784	175,686	**71,569**
28	150,568	50,14	229,50	109,972	146,470	179,357	**133,371**
29	176,331	79,50	280,01	116,215	181,793	200,507	**162,162**
30	255,584	101,06	280,00	221,403	293,130	178,935	267,229
31	350,442	337,17	519,09	248,533	366,208	181,916	328,407
32	481,45	322,70	501,15	364,560	516,903	178,443	465,787
33	491,849	280,05	465,38	626,663	901,918	185,327	811,096
34	927,548	759,53	958,51	603,882	932,726	198,980	832,772
35	968,834	943,11	1156,81	712,720	1152,694	213,702	1026,227

Варто зазначити, що в лінійних моделях у деяких випадках трапляються від'ємні значення як у довірчих інтервалах, так і в інтервалах передбачення, що є неприйнятним.

Таблиця 2.5 демонструє оцінку якості лінійних і нелінійних регресійних моделей, які використовуються для визначення розміру модулів електронної комерції на основі метрик, зібраних із PHP-проектів.

Таблиця 2.5 – Порівняння оцінок лінійної та нелінійної моделей

Оцінки якості моделі	Модель	
	Лінійна	Нелінійна
R^2	0,972	0,943
$MMRE$	2,499	0,188
$PRED(0{,}25)$	0,4	0,714

Як видно з таблиці, нелінійна модель забезпечує значно кращі результати за метриками $MMRE$ і $PRED(0{,}25)$, порівняно з лінійною. Значення R^2 високе для обох моделей, але точність оцінок у нелінійній моделі є вищою. Проте, у нелінійній моделі $PRED(0{,}25)$ дещо менше задовільного.

Для аналізу також було проведено порівняння з іншими існуючими нелінійними моделями, розробленими для PHP-фреймворків. Дані наведено у таблиці 2.6, де використані ті самі метрики-предиктори, що й у нашому дослідженні.

Таблиця 2.6 – Порівняння нелінійних регресійних моделей

Оцінки якості моделі	Модель			
	e-commerce	CakePHP	Yii	Symfony
R^2	0,943	0,930	0,848	0,852
$MMRE$	0,188	0,265	0,303	0,230
$PRED(0{,}25)$	0,714	0,514	0,371	0,543

З результатів видно, що використання моделей, створених для інших типів проектів (зокрема, PHP-фреймворків), дає гірші показники R^2, $MMRE$ та

$PRED(0{,}25)$. Це підкреслює важливість розробки регресійних моделей, спеціально адаптованих до конкретного фреймворку, домену, типу проектів програмного забезпечення.

2.3 Висновки до розділу 2

У цьому розділі було розроблено нелінійну регресійну модель для оцінювання розміру модулів для електронної комерції на мові PHP з використанням нормалізуючого перетворення log10.

Основні етапи роботи:

– Проведено пошук проектів модулів для електронної комерції на платформі GitHub. Використання інструменту PhpMetrics дозволило отримати метрики діаграми класів, необхідні для побудови моделі.

– Перевірено метрики на мультиколінеарність, яка не була виявлена. Крім того, розрахунок квадрату відстані Махаланобіса підтвердив негаусівський характер розподілу даних.

– Видалено викиди із початкового набору даних за допомогою нормалізації та аналізу квадрату відстані Махаланобіса. У результаті було виключено три проекти.

– Побудовано лінійну і нелінійну регресійні моделі для очищених даних. Лінійна модель була створена для порівняння за припущенням про нормальність розподілу. Результати порівняння свідчать про перевагу нелінійної моделі.

– Для обох моделей побудовано довірчі інтервали та інтервали передбачення. Нелінійна модель продемонструвала менші ширини інтервалів, що підвищує її достовірність.

– Проведено порівняння побудованої нелінійної моделі з існуючими моделями для фреймворків CakePHP, Yii, Symfony. Результати показують значно кращі показники якості для побудованої моделі.

3 ПРОЄКТУВАННЯ ТА СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЗНАЧЕННЯ РОЗМІРУ МОДУЛІВ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА PHP

3.1 Постановка задачі на розробку проекту ПЗ для оцінювання розміру модулів електронної комерції на PHP

Одним із ключових завдань цієї роботи є створення ПЗ, яке дозволить кінцевому користувачеві оцінювати розмір модулів електронної комерції на PHP. Програму планується зробити сумісною з найпоширенішими сучасними операційними системами. Це вимагає ретельного підбору мови програмування, бібліотек та інших інструментів.

Оптимальним рішенням у цьому контексті є Python. Ця мова з відкритим кодом підтримує строгість типізації та забезпечує багатий функціонал завдяки бібліотекам NumPy, SciPy, Pandas, які полегшують математичні обчислення. Для створення графічного інтерфейсу пропонується використовувати бібліотеку Qt, яка є гнучким інструментом для побудови кросплатформних застосунків. Крім того, Python не має обмежень для комерційного використання, що робить його ідеальним для реалізації подібного проєкту.

У межах цієї роботи розробка буде виконана відповідно до технічного завдання, представленого у додатку А. Також буде створено ескізний, технічний і робочий проєкти, що охоплюють усі ключові аспекти функціональності програмного забезпечення.

3.2 Розробка ескізного проекту ПЗ для оцінювання розміру модулів електронної комерції на PHP

У межах ескізного проекту здійснюється концептуальне моделювання системи, що розробляється, зокрема побудова діаграми варіантів використання й діаграми діяльності.

3.2.1 Розробка сценарії використання ПЗ для оцінювання розміру модулів електронної комерції на PHP

Діаграма прецедентів використання – це схематичне зображення функціональності системи з точки зору користувачів. Вона є важливим інструментом для збору вимог на ранніх етапах розробки, оскільки описує, що система повинна робити, не вдаючись у деталі реалізації.

Елементи діаграми:

– Актори (користувачі або зовнішні системи), які взаємодіють із системою.

– Варіанти використання (сценарії функцій), які зображуються у вигляді овалів.

– Зв’язки між акторами та варіантами використання, які показують, хто і як використовує систему.

Таблиця 3.1 – Визначені прецеденти

Основні актори	Найменування	Формулювання
1	2	3
Користувач	Задати дані з метрик e-commerce модулів	Визначає шлях до файлу з метриками e-commerce модулів
Користувач	Задати кількість класів	Задає кількість класів у модулів
Користувач	Задати середню кількість методів на кожен клас	Задає середню кількість методів для класів у модулів
Користувач	Задати глибину дерева наслідування	Задає глибину наслідування у модулів
Користувач	Задати довірчу ймовірність	Задає значення довірчої ймовірності

Продовження таблиці 3.1

1	2	3
Користувач	Нормалізувати метрики	Отримує нормалізовані дані за логарифмічним перетворенням
Користувач	Побудувати нелінійну регресійну модель	Здійснює побудову та отримує результати моделі
Користувач	Оцінити розмір застосунку	Визначає розмір системи, довірчий інтервал та інтервал передбачення
Користувач	Оцінити інтервал передбачення	Генерує інтервал передбачення для застосунку
Користувач	Оцінити довірчий інтервал	Визначає довірчий інтервал на основі розрахунків

Діаграму сценаріїв використання показано на рисунку 3.1.

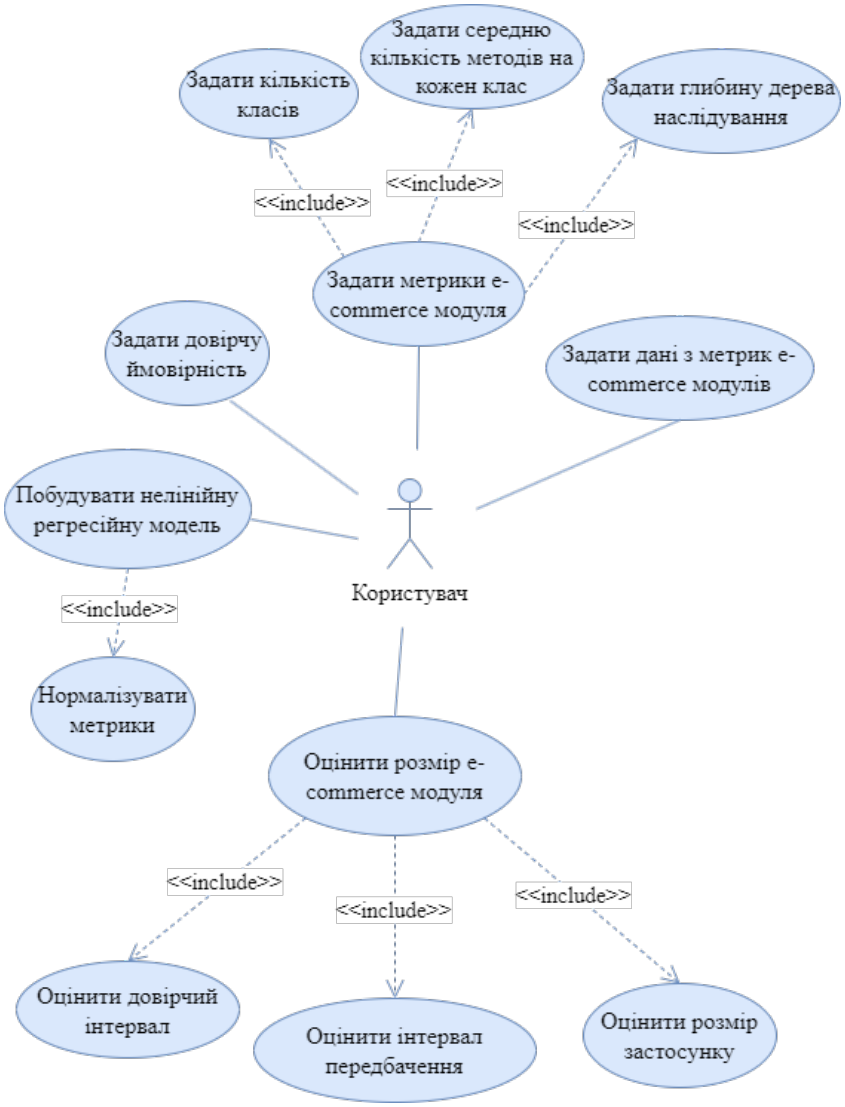

Рисунок 3.1 – Діаграма сценаріїв використання

3.2.2 Модель сценарію виконання ПЗ для оцінювання розміру модулів електронної комерції на PHP з застосування діаграми діяльності

Діаграма діяльності – це графічний інструмент, який моделює послідовність виконання дій у рамках певної системи або процесу. Вона використовується для відображення алгоритмів, бізнес-процесів чи робочих потоків, включаючи паралельні виконання завдань і прийняття рішень. Основні елементи такої діаграми включають:

– Дії (Activity) – конкретні задачі, що виконуються.

– Рішення (Decision) – вузли, де визначається подальший напрямок потоку на основі умов.

– Потоки переходів (Flows) – стрілки, які показують послідовність виконання завдань.

– Стан початку та завершення (Start/End States) – позначають початок і кінець процесу.

Ця діаграма є корисною для створення чіткої структури роботи системи, забезпечення узгодженості між розробниками, тестувальниками та іншими учасниками проекту, а також для ефективного планування ресурсів. Її зручно використовувати для виявлення потенційних проблем у процесах ще на етапі проектування.

Діаграму діяльності показано на рисунку 3.2.

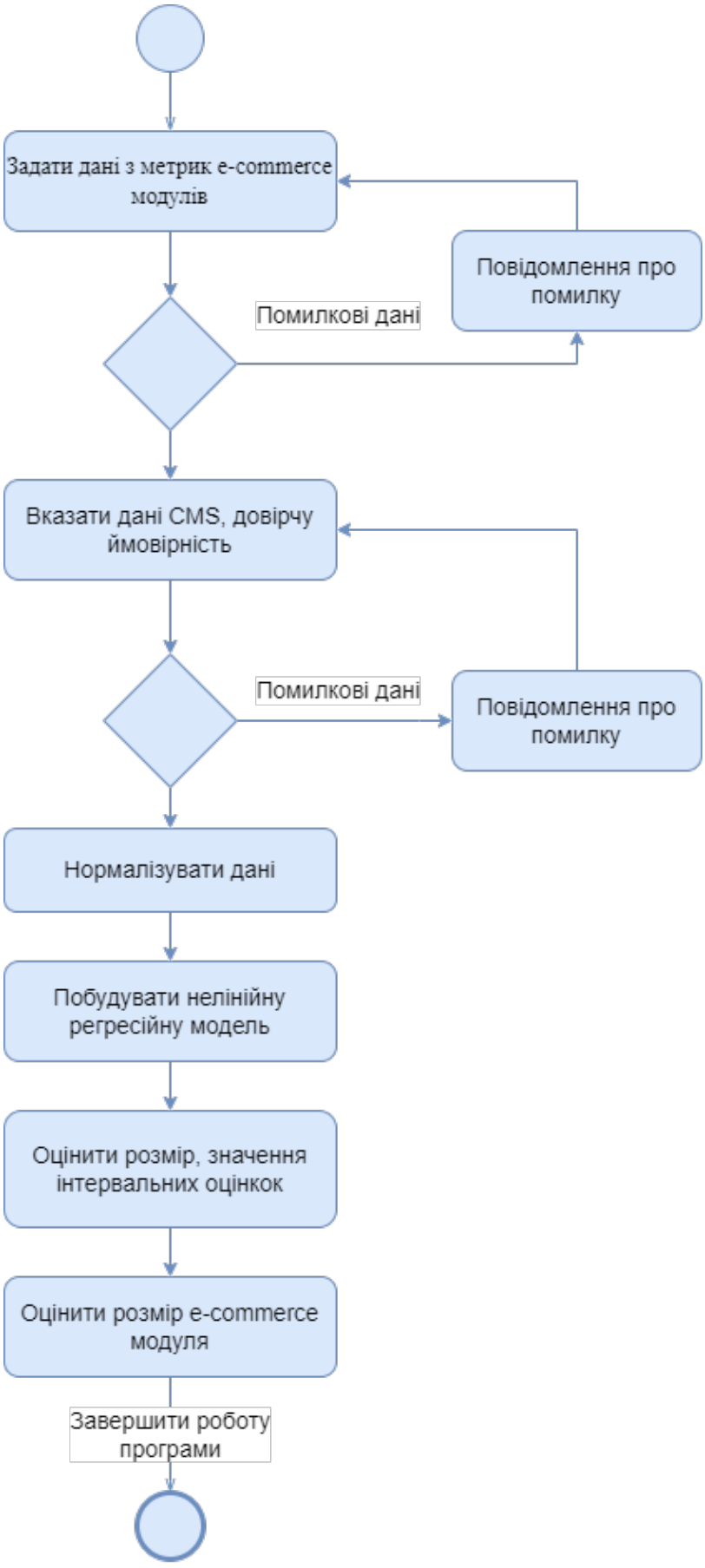

Рисунок 3.2 – Діаграма діяльності

3.3 Розробка технічного проекту ПЗ для оцінювання розміру модулів електронної комерції на PHP

3.3.1 Представлення класів ПЗ для оцінювання розміру модулів електронної комерції на PHP

Діаграма класів у UML моделює статичну структуру системи, відображаючи її основні компоненти – класи, їхні атрибути та методи – і типи зв’язків між ними. Вона є базовим інструментом для об’єктно-орієнтованого аналізу та проєктування, забезпечуючи візуалізацію внутрішньої організації системи.

На діаграмі показуються:

– Класи, що представляють ключові об’єкти системи.

– Атрибути, які визначають властивості класів.

– Методи (операції), що описують поведінку класів.

– Зв’язки (асоціації, агрегації, композиції), які ілюструють, як класи взаємодіють між собою.

– Успадкування, що описує ієрархічні відносини.

Ця діаграма широко використовується для планування архітектури системи, визначення залежностей між модулями та документації проекту. Вона дозволяє розробникам і бізнес-аналітикам краще зрозуміти структуру системи та забезпечити узгодженість під час командної роботи.

Діаграму класів показано на рисунку 3.3.

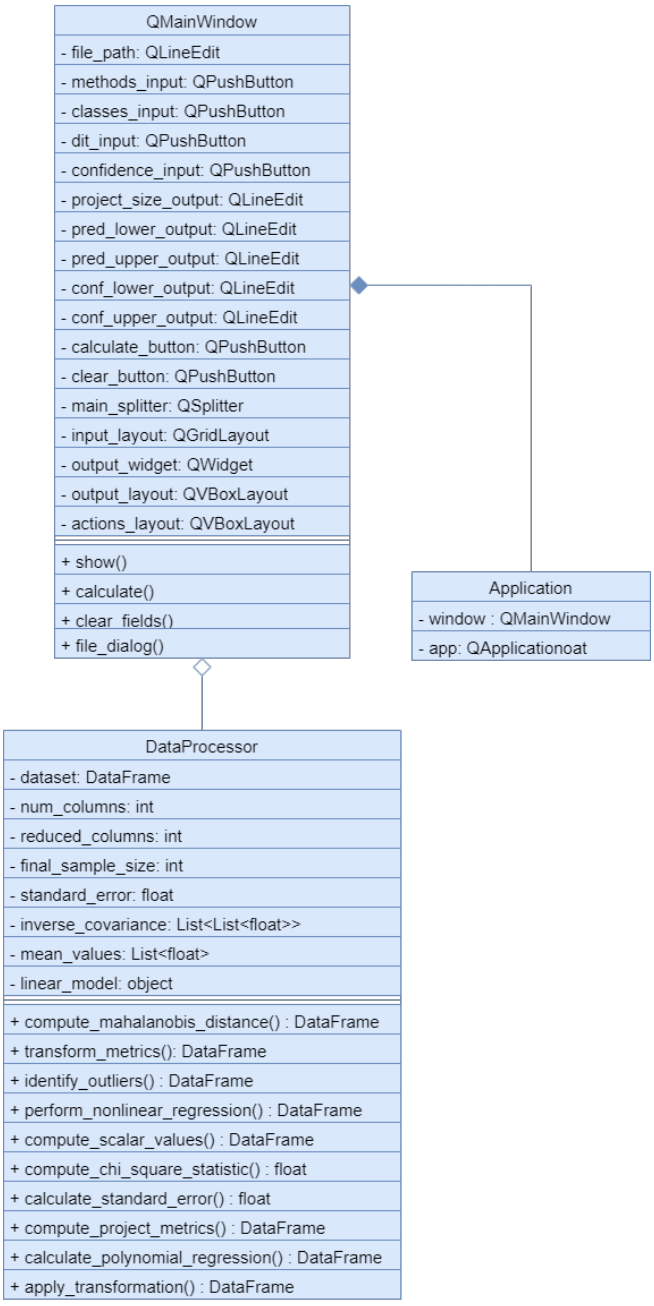

Рисунок 3.3 – Діаграма класів

3.3.2 Специфікації класів ПЗ для оцінювання розміру модулів електронної комерції на PHP

– QMainWindow – клас, який реалізує головне вікно додатку.

– QtWidget – базовий клас бібліотеки Qt, що слугує основою для елементів графічного інтерфейсу.

– QLineEdit – клас, який відповідає за текстові поля в графічному інтерфейсі.

– QValidator – механізм валідації даних у реальному часі, який блокує введення некоректної інформації.

– Applacation – ключовий клас програми, який управляє її життєвим циклом.

– DataProcessor – клас для створення регресійних моделей, оцінювання веб-застосунків і розрахунку інтервальних оцінок.

3.3.3 Моделювання алгоритмів виконання ПЗ для оцінювання розміру модулів електронної комерції на PHP

Діаграма послідовностей у UML описує динаміку взаємодії об’єктів у системі. Вона моделює процеси, показуючи порядок і залежність викликів методів або обміну повідомленнями між об’єктами в часі. Основні складові:

– Учасники (актори та об’єкти), які беруть участь у взаємодії.

– Повідомлення, що показують, які дії ініціюють об’єкти або актори.

– Часовий вимір, що дозволяє зрозуміти, в якому порядку відбуваються виклики.

– Життєва лінія, яка позначає існування об’єкта під час взаємодії.

Ця діаграма є надзвичайно корисною для деталізації функціональних вимог, аналізу сценаріїв використання та виявлення можливих проблем у логіці роботи системи.

Послідовність виконання прецеденту «Ввести дані e-commerce модуля» показана на рисунку 3.4.

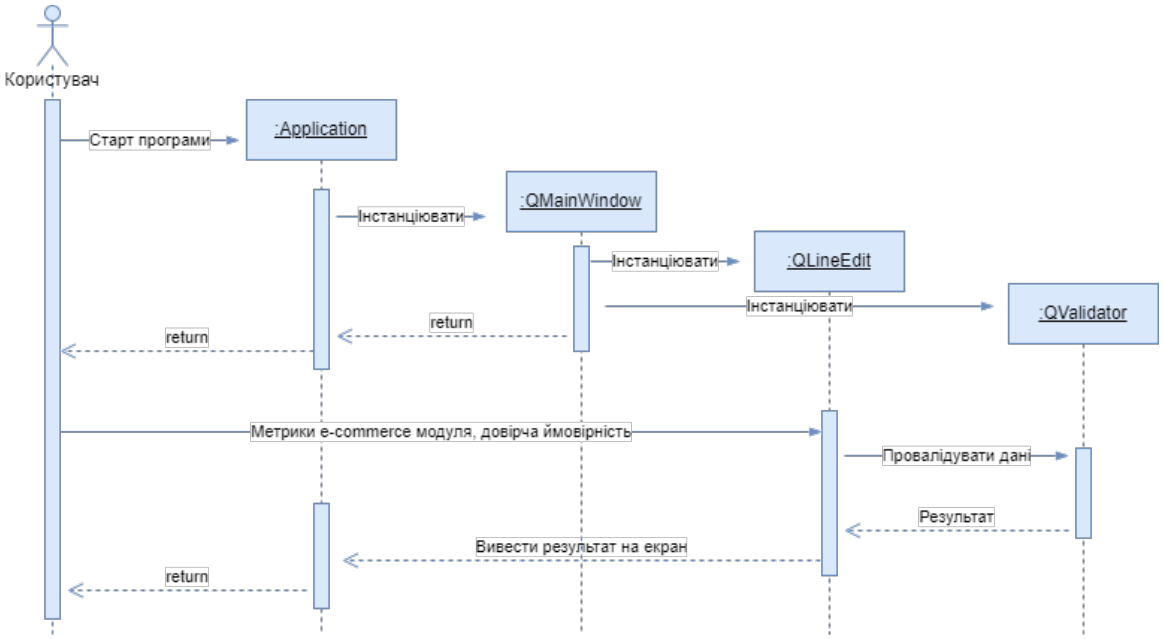

Рисунок 3.4 – Послідовність виконання прецеденту «Задати метрики e-commerce модуля»

Послідовність виконання прецеденту «Виконати оцінювання розміру e-commerce модуля» показана на рисунку 3.5.

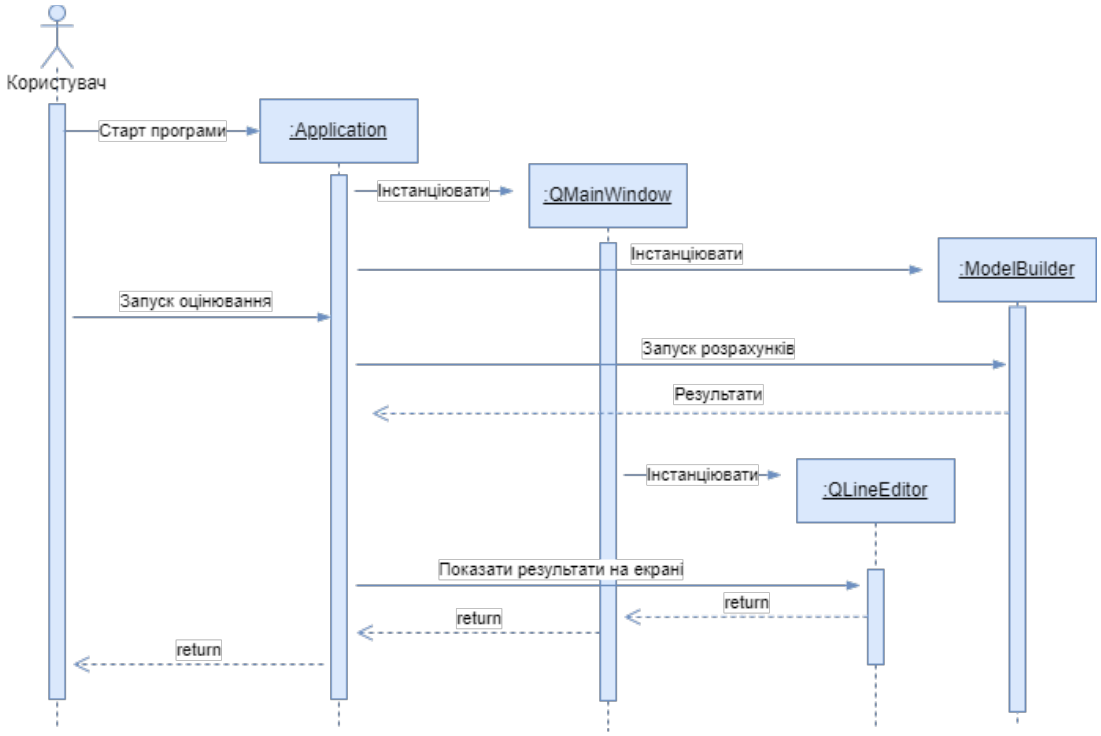

Рисунок 3.5 – Послідовність виконання прецеденту «Оцінити розмір e-commerce модуля»

3.4 Розробка робочого проекту ПЗ для оцінювання розміру модулів електронної комерції на PHP

3.4.1 Кодування ПЗ для оцінювання розміру модулів електронної комерції на PHP

У межах робочого проєкту створено програму, яка дозволяє оцінювати розмір модулів для електронної комерції на PHP, вимірюючи його в тисячах рядків коду. Було виконано тестування програми та її випробування. Розробка виконана за допомогою мови Python, а для створення графічного інтерфейсу застосовано бібліотеку Qt, яка забезпечує сумісність із Python. Повний вихідний код програми представлено у додатку Г.

3.4.2 Тестування ПЗ для оцінювання розміру модулів електронної комерції на PHP

Еквівалентне розбиття – це методика тестування, що дозволяє зменшити кількість тестових прикладів шляхом групування вхідних даних у класи, які повинні давати однакові результати. Вважається, що якщо один приклад з класу пройшов тест успішно, то й інші приклади цього класу працюватимуть коректно.

Цей підхід використовується для перевірки:

– Валідації вхідних даних.

– Коректності роботи алгоритмів із різними наборами значень.

Еквівалентне розбиття спрощує тестування, допомагаючи сфокусуватися на ключових областях програми.

Для перевірки функціоналу розроблено та протестовано сценарії використання, зокрема: введення кількості класів, кількості методів на кожен клас, а також глибини дерева наслідування.

Виявлені класи еквівалентності описані у таблицях 3.2 та 3.5, а результати тестів – у таблицях 3.3, 3.4, 3.6, та 3.7.

Таблиця 3.2 показує виявлені класи еквівалентності прецеденту «Задати кількість класів»

Таблиця 3.2 – Виявлені класи еквівалентності

Вхідні данні	Правильні класи	Неправильні класи
Значення кількості класів	Натуральне число (1)	Дійсне число (2). Від’ємне число (3). Не число (4) Без значення (5).

Таблиця 3.3 показує тести правильних класів цього прецеденту

Таблиця 3.3 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	44	Подальше виконання програми	Пройдено

Таблиця 3.4 показує тести неправильних класів цього прецеденту

Таблиця 3.4 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	2	11,32	Помилка на екрані	Пройдено
3	3	-100,12	Помилка на екрані	Пройдено
4	4	Слово	Помилка на екрані	Пройдено
5	5	-	Заклик заповнити поле	Пройдено

Таблиця 3.2 показує виявлені класи еквівалентності прецеденту «Задати середню кількість методів на кожен клас»

Таблиця 3.5 – Виявлені класи еквівалентності

Вхідні данні	Правильні класи	Неправильні класи
Середня кількість методів на кожен клас (глибина дерева наслідування)	Дійсне додатне число (1)	Від’ємне число (2). Не число (3) Без значення (4).

Таблиця 3.6 показує тести правильних класів цього прецеденту

Таблиця 3.6 – Тести правильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	4,44	Подальше виконання програми	Пройдено

Таблиця 3.7 показує тести неправильних класів цього прецеденту

Таблиця 3.7 – Тести неправильних класів еквівалентності

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
2	2	-100,12	Помилка на екрані	Пройдено
3	3	Слово	Помилка на екрані	Пройдено
4	4	-	Заклик заповнити поле	Пройдено

Тестування введення глибини дерева наслідування виконується аналогічно сценарію для середньої кількості методів. Процес перевірки введення довірчої ймовірності відбувається подібно до інших варіантів використання, за винятком того, що це значення має бути дійсним числом у діапазоні від 0 до 1.

Результати тестів свідчать про відповідність роботи програми заявленим вимогам.

3.4.3 Випробування ПЗ для оцінювання розміру модулів електронної комерції на PHP

На основі затвердженої методики випробувань, представленої у Додатку Б, було виконано оцінювання розміру модуля для електронної комерції на мові PHP. У процесі тестування розглянуто такі функціональні можливості:

– введення метрик e-commerce модуля;

– введення даних для оцінювання розміру e-commerce модуля;

– нормалізація даних;

– створення нелінійної регресійної моделі;

– оцінка розміру e-commerce модуля з урахуванням довірчих інтервалів та інтервалів передбачення.

Результати випробувань представлено на рисунках 3.6 та 3.7, що підтверджує відповідність програмного забезпечення заявленим вимогам.

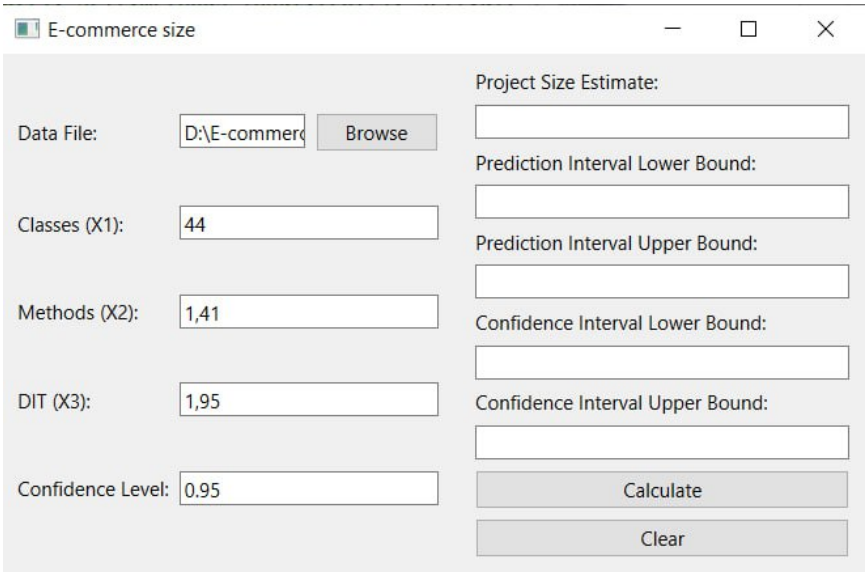

Рисунок 3.6 – Вікно програми перед виконанням оцінювання

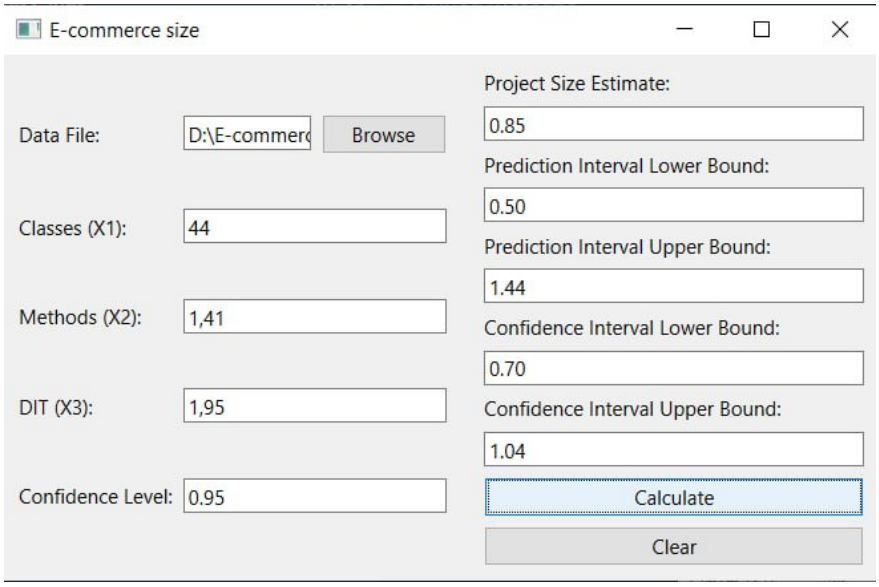

Рисунок 3.8 – Вікно програми після виконання оцінювання

Правильна робота програми підтверджує відповідність вимогам, висунутим у технічному завданні.

3.5 Висновки до розділу 3

Розділ 3 присвячено розробці ПЗ для оцінювання розміру модулів електронної комерції на PHP. Виконано аналіз вимог і побудовано концептуальні моделі системи, включаючи діаграми варіантів використання, діяльності та послідовності. За допомогою Python та бібліотеки Qt створено програму, яка успішно пройшла тестування із застосуванням методики еквівалентного розбиття. Випробування програми підтвердили її здатність виконувати оцінювання розміру модулів із довірчими та прогнозними інтервалами відповідно до заданих вимог.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ І ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОДУЛІВ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА МОВІ PHP

Вступ

Доцільність створення програмного забезпечення може бути підтверджена з економічної точки зору шляхом оцінки його собівартості та передбачуваного прибутку. Основним критерієм оцінки економічної доцільності витрат на розробку є річний економічний ефект. Обґрунтування економічної доцільності базується на розрахунках собівартості програмного продукту та прибутку, який очікується від його продажу. Річний економічний ефект є ключовим показником у цій оцінці.

4.1 Розрахунок витрат на створення й експлуатацію програмного забезпечення

Витрати на розробку програмного забезпечення складаються з витрат на заробітну плату розробника, на амортизацію та експлуатацію ЕОМ, на якій виконується розробка, на засоби розробки та витрати на матеріали і комплектуючі. Розробка програмного забезпечення виконується спеціалістом, місячний оклад якого складає 21000 грн. Додаткова заробітна плата складає 20% від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 24200 грн./міс , а вартість сучасної ПЗВМ складає 25000 грн. Вартість кіловат-години електроенергії дорівнює 4,32 грн. Витрати на допоміжні матеріали наведені в таблиці 4.1

Таблиця 4.1 – Витрати на допоміжні матеріали

№	Пункти витрат	Сума, грн.
1	Папір для принтера	350
2	Картридж та тонер для принтера	450
3	Використання флеш-накопичувача	300
4	Непередбачувані витрати	2000
	Разом	3100

Вартість розробки системи розраховується за формулою:

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

де Т – тривалість розробки у місяцях

$З_{зп}$ – основна і додаткова заробітна плата, грн.;

$З_{зс}$ – відрахування на соціальні заходи (38% від основної і додаткової заробітної плати), грн.;

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

$З_{е}$ – витрати на електроенергію, грн.

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

$З_{е}$ при витратах в 0,5 кВт з тривалістю роботи на місяць 22*8 = 176 годин та вартості електроенергії 4,32 грн. становить

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

Представимо всі поточні витрати на розробку програмного забезпечення в таблиці 4.2

Таблиця 4.2 – Витрати на розробку програмного забезпечення

№	Пункти витрат	Одиниця	Кількість
1	Тривалість розробки (Т)	Міс.	2
2	Основна та додаткова заробітна плата (Ззп)	Грн.	24200
3	Відрахування на соціальні витрати (Ззс)	Грн.	9136
4	Загальногосподарські витрати (Ззг)	Грн.	2420
5	Витрати на допоміжні матеріали (Зм)	Грн.	3100
6	Витрати на електроенергію *(Зе)	Грн.	380,16

За формулою (4.1) виконаємо розрахунок вартості програми:

$$С_{пр} = (24200 + 9136 + 2420 + 380{,}16) * 2 + 3100 = 75327{,}32 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60% балансової вартості в рік:

$$А_{об} = 25000 * 0{,}6 = 15000 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали визначаються в розмірі 5% вартості основного устаткування:

$$В_{м} = 23000 * 0{,}05 = 1250 \text{ грн.}$$

Річний обсяг робіт ПК у годинах можна визначити як

$$Ф_{м} = 264{,}5 * Т_{з}, \quad (4.2)$$

де Тз – середнє навантаження на устаткування в місяць (6 годин), 264,5 – середня кількість робочих днів на рік.

Отже, річний обсяг роботи ПК складає:

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

Витрати на електроенергію Зе при 1578 годинах роботи устаткування на рік становлять:

$$З_{е} = 1587 * 4{,}32 = 6555{,}84$$

Експлуатаційні витрати на ПК на рік становлять:

$$З_{зр} = 15000 + 1250 + 6555{,}84 = 22805{,}84$$

Таким чином витрати на розробку та експлуатацію програмного забезпечення в перший рік становлять:

$$З_{р} = 75327{,}32 + 22805{,}84 = 98133{,}16$$

4.2 Економічна ефективність розробки та впровадження програмного забезпечення

Визначити пряму економічну ефективність можна, ґрунтуючись на тому, що впровадження програмного забезпечення вивільняє одного працівника (за експертною оцінкою фахівців підприємства). Зарплата одного працівника в рік складає

$$З = 21000 * 12 * 1 = 252000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$E_{рік} = \Delta C_n - E_n * k, \quad (4.3)$$

де ΔC_n – кошти, що буде вивільнено при впровадженні програмного забезпечення (252000) без експлуатаційних витрат (22805,84), тобто, 229194,16 грн.;

E_n – коефіцієнт ефективності (такий же як й коефіцієнт амортизації – 0,6)

k – одноразові витрати на впровадження програмного забезпечення (75327,32 грн.).

$$E_{рік} = 229194,16 - 0,6 * 75327,32 = 229194,16 - 45196,39 = 183997,77$$

Термін, за який програмне забезпечення окупиться наступний:

$$Т = \frac{k}{\Delta C_n} = \frac{75327,32}{229194,16} = 0,33 \text{ року}$$

Отже термін, за який програмне забезпечення окупиться 4 місяці.

4.3 Висновки до розділу 4

У цьому розділі виконано розрахунки, що стосуються створення та впровадження програмного забезпечення для оцінювання розміру модулів для електронної комерції на мові PHP. Після цього було визначено економічну

ефективність і термін окупності розробленого продукту. За результатами розрахунків, програмне забезпечення окупиться протягом 4 місяців у перший рік використання. Таким чином, розробка програмного забезпечення є економічно вигідною.

5 ОХОРОНА ПРАЦІ

Охорона праці – це комплекс заходів і засобів правового, соціально-економічного, організаційного, технічного та гігієнічного спрямування, метою якого є збереження здоров’я, працездатності та життя працівників у процесі їхньої професійної діяльності. Система охоплює:

– оцінку шкідливих і небезпечних факторів;

– впровадження методів усунення ризиків і забезпечення безпечних умов праці;

– дотримання правил техніки безпеки;

– спеціальні заходи для жінок, молоді та працівників зі зниженою працездатністю;

– норми, адаптовані до умов роботи на шкідливих і небезпечних виробництвах.

Безпечність праці залежить від численних факторів, насамперед від професіоналізму та відповідальності керівників. Система стандартів безпеки праці є важливим елементом, вона спрямована на попередження травм і професійних хвороб.

Закон України "Про охорону праці" встановлює правові механізми забезпечення безпечних умов роботи, регулює взаємодію між роботодавцем і працівником і визначає порядок організації заходів із безпеки праці.

Удосконалення умов праці позитивно позначається на економіці підприємства, підвищує продуктивність праці, знижує собівартість і сприяє покращенню якості продукції.

Серед соціальних результатів виділяють збереження здоров’я працівників, підвищення задоволеності роботою, зростання престижу професій та виробничої активності.

5.1 Аналіз шкідливих та небезпечних факторів в офісі фірми

Під час виконання службових обов’язків працівники відділу стикаються з небезпечними та шкідливими факторами, зокрема:

– поганим освітленням робочих зон;

– шумом і вібрацією від роботи офісного обладнання;

– підвищеним ризиком пожеж через скупчення паперової документації;

– загрозою ураження струмом через високу напругу в мережі;

– електромагнітним випромінюванням, що супроводжує роботу комп’ютерів;

– незадовільними параметрами мікроклімату, такими як температура, вологість чи рух повітря.

Робота вимагає зорової концентрації, тому у приміщенні організовано освітлення трьох видів: природне, комбіноване та штучне. Природне освітлення через вікна забезпечує комфорт у денний час, комбіноване додає штучні джерела за поганих погодних умов, а ввечері застосовується лише штучне світло.

Головним джерелом шуму є вентилятори та комп’ютерна техніка. Для його зниження використовуються спеціальні звукопоглинаючі матеріали, такі як мінеральна вата.

Вібрація, створювана технікою, не перевищує норм. У приміщенні експлуатуються два комп’ютери та один струменевий принтер.

Через наявність легкозаймистих меблів зростає ризик швидкого поширення вогню у разі пожежі. Електроприлади обладнані заземленням, що знижує ймовірність ураження струмом.

Випромінювання від моніторів спричиняє накопичення пилу, який може викликати алергічні реакції. Електромагнітне поле також може негативно позначитися на продуктивності праці.

Мікроклімат у відділі підтримується системами вентиляції та опалення, забезпечуючи оптимальні умови для роботи техніки та комфорту співробітників.

5.2 Заходи щодо зменшення впливу шкідливих факторів роботи за персональним комп’ютером

Для зменшення негативного впливу шкідливих факторів у приміщенні необхідно впровадити низку заходів. Для забезпечення правильного освітлення робочих місць пропонується використовувати світильники типу УПМ із двома лампами в кожному, розташованими двома рядами по три світильники й одним додатковим посередині.

Щоб знизити рівень шуму, можна застосувати звукоізоляцію, малошумне обладнання та оптимізувати розташування джерел шуму.

Для зменшення ризику пожежі доцільно автоматизувати документообіг, залишаючи більшість інформації в електронному форматі. Безпечну експлуатацію електроприладів можна досягти через підвищення технічної грамотності співробітників і дотримання правил технічної експлуатації. Захист від електромагнітного випромінювання комп’ютерів забезпечується встановленням екранів, а для покращення якості повітря варто обладнати приміщення системою кондиціонування. Усі ці заходи разом дозволять створити комфортні умови для роботи, підвищуючи продуктивність.

6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Захист навколишнього середовища охоплює низку дій, спрямованих на обмеження негативного впливу людської діяльності на природу. Державне регулювання у цій сфері здійснюють Кабінет Міністрів України та органи місцевої влади. Міністерство охорони навколишнього середовища і його місцеві представництва є спеціалізованими установами з контролю за екологічними питаннями.

Серед головних правових актів, що регулюють охорону природи, – Закони України: «Про охорону навколишнього природного середовища» (1991 р.), «Про охорону атмосферного повітря» (1992 р.), «Про природно-заповідний фонд» (1992 р.) та інші.

Екологічна охорона охоплює правові механізми, які закріплюють екологічні принципи, матеріальні стимули для підприємств і впровадження інженерних рішень у природоохоронній сфері.

Завдання охорони довкілля включають: зменшення викидів, створення заповідних зон, захист рідкісних видів тварин, боротьбу з несанкціонованими звалищами.

Особливу увагу приділяють проблемам забруднення атмосфери. Шкідливі гази викликають озонові «дірки», посилення ультрафіолетового випромінювання та розвиток парникового ефекту, що веде до танення льодовиків, підвищення рівня океану і зміни клімату.

Із розвитком техніки зросли викиди техногенних речовин, таких як сполуки сірки, азоту і галогенів, що ускладнює екологічну ситуацію. Необхідно зменшувати ці викиди для збереження якості повітря.

6.1 Забруднення навколишнього середовища

Комп'ютерна компанія, яка займається розробкою та супроводженням програмного забезпечення, хоч і не є виробничою, все ж може мати джерела негативного впливу на довкілля.

До основних джерел забруднення належать:

– дизельна електростанція;

– котли на твердому паливі;

– побутове сміття; е

– лектромагнітне випромінювання;

– техніка, що вийшла з ладу.

Для забезпечення безперебійної роботи офісу компанія використовує дизельну електростанцію, яка функціонує у випадках аварійного чи резервного вимкнення електроенергії. Однак її використання спричиняє викиди продуктів горіння палива та шум.

Системи опалення приміщень працюють на деревних пелетах, що мають менший рівень викидів, ніж вугілля, однак не є повністю безпечними для екології.

Робота компанії також призводить до утворення побутових відходів – як рідких (стічні води, нечистоти), так і твердих (пластик, скло, папір, харчові відходи). Організація їх утилізації є важливим питанням.

Окремої уваги потребує утилізація застарілої оргтехніки. Зростання кількості комп’ютерів і периферійного обладнання робить це питання актуальним. Обладнання, що більше не використовується, містить дорогоцінні метали та небезпечні речовини. Неправильна утилізація може спричинити забруднення довкілля.

Офісна техніка також є джерелом електромагнітного випромінювання. Постійна робота комп’ютерів, серверів і мікрохвильових печей може негативно впливати на здоров’я співробітників, викликаючи дискомфорт та хронічну втому.

6.2 Розробка заходів щодо зменшення забруднення

Раціональне використання природних ресурсів, захист довкілля та екологічна безпека життєдіяльності є ключовими умовами для сталого розвитку України. Основні екологічні відносини регулюються Законом України «Про охорону навколишнього природного середовища», а також спеціальними законами, такими як земельне, водне, лісове законодавство, норми охорони повітря, надр, флори і фауни. Задля зменшення рівня забруднення навколишнього середовища необхідно впроваджувати наступні заходи:

– Ефективне очищення викидів – на великих енергоблоках використовуються електрофільтри для очищення димових газів, а на котлах середньої потужності – мокрі золоуловлювачі. Уловлювання частинок золи відбувається за рахунок відцентрової сили на водяній плівці, що забезпечує ефективність до 90%.

– Використання екологічного палива – природний газ замість деревних пелетів зменшує рівень шкідливих викидів. Водночас слід враховувати економічну доцільність такого рішення.

– Поводження з побутовими відходами – енергетичне спалювання деяких відходів, переробка макулатури, скла й пластику, повторне використання тари, а також скорочення кількості пакування є ефективними способами зменшення навантаження на екосистему.

– Обмеження електромагнітного впливу – варто вимикати пристрої, якими не користуються, і регулярно виходити на прогулянки. Рівень випромінювання мікрохвильових печей потрібно періодично перевіряти.

– Переробка техніки – утилізація старої оргтехніки та комп’ютерів дозволяє повторно використовувати до 95% матеріалів, мінімізуючи кількість сміття, що потрапляє на полігони.

Будь-яка комп’ютерна техніка чи оргтехніка може бути перероблена та використана повторно. При правильній утилізації до 95% відходів повертаються у виробництво, а лише 5% потрапляють на сміттєзвалища або заводи з переробки

побутового сміття. На фабриках, де здійснюється переробка, частка ручної та автоматичної роботи залежить від типу техніки. Так, для моніторів це співвідношення становить близько 50 на 50 через складність розбирання кінескопів. Натомість для системних блоків і оргтехніки частка автоматизації значно вища.

Уперше компанія HP розпочала переробку старої техніки у 1981 році. На сьогодні вона має глобальну інфраструктуру збирання та утилізації ПК і оргтехніки у 50 країнах. Щороку компанія переробляє близько 2,5 мільйона одиниць техніки. Лише у 2007 році HP утилізувала понад 100 тисяч тонн списаного обладнання та витратних матеріалів, що на 50% більше, ніж роком раніше.

Переробка комп'ютерної техніки за КВЕД потребує підготовчих процедур:

– Форматування жорстких дисків на низькому рівні.

– Очищення картриджів від залишків тонера, що містить шкідливі хімічні речовини. Електронні відходи розбираються, сортуються та обробляються. Пластикові, металеві та скляні частини оргтехніки переробляють окремо.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи було вирішено завдання побудови нелінійної регресійної моделі для оцінювання розміру модулів для електронної комерції на мові PHP, а саме:

– Проведено аналіз існуючих методів оцінювання розміру веб-застосунків на PHP. Виявлено, що найчастіше використовуються нелінійні регресійні моделі, які базуються на даних із діаграм класів. Обґрунтовано потребу у створенні такої моделі для модулів електронної комерції на мові PHP.

– Виконано пошук проектів модулів для електронної комерції на платформі GitHub, зібрано метрики діаграми класів за допомогою PhpMetrics: кількість класів, середню кількість методів на клас, глибину дерева наслідування.

– Проведено тестування метрик на нормальність розподілу. Виявлено, що дані мають негаусівський характер і містять викиди. Викиди були видалені за допомогою перетворення log10 і аналізу квадрату відстані Махаланобіса.

– Побудовано трьохфакторну лінійну та нелінійну регресійні моделі. Нелінійна модель перевершила лінійну за всіма ключовими показниками якості: R^2, $MMRE$ та $PRED(0{,}25)$.

– Розраховано довірчі інтервали та інтервали передбачення для обох моделей. Нелінійна модель забезпечує менші ширини інтервали, що свідчить про її перевагу на лінійною.

– Проведено порівняння побудованої нелінійної моделі з існуючими моделями для фреймворків CakePHP, Yii, Symfony. Результати показують значно кращі показники якості для побудованої моделі.

– Проведено розрахунок економічної ефективності щодо розробки програмного забезпечення для оцінювання розміру модулів для електронної комерції на мові PHP, розрахунки показують доцільність розробки.

– Розроблено розділи з охорони праці та охорони навколишнього середовища.

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: 07.12.2024).

2. Boehm B.W. Software Cost Estimation with COCOMO II. Prentice Hall, 2000. 544 p.

3. Fenton N., Bieman J. Software Metrics. CRC Press, 2014. URL: https://doi.org/10.1201/b17461 (date of access: 07.12.2024).

4. Zifen Yang. An improved software size estimation method based on object-oriented approach. *2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM)*, Kuala Lumpur, Malaysia, 24–27 June 2012. 2012. URL: https://doi.org/10.1109/eeesym.2012.6258733 (date of access: 17.11.2024).

5. Trendowicz A., Jeffery R. Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success. Springer, 2014.

6. Usage statistics of PHP for websites. URL: https://w3techs.com/technologies/details/pl-php. (дата звернення: 12.11.2024).

7. Prykhodko N.V., Prykhodko S.B. The non-linear regression model to estimate the software size of open source Java-based systems. *Radio Electronics, Computer Science*, Control. 2018. No. 3. URL: https://doi.org/10.15588/1607-3274-2018-3-17 (date of access: 17.11.2024)

8. Prykhodko N.V., Prykhodko S.B. Non-linear regression model to estimate the software size of vb-based information systems. *Information Technology And Computer Engineering*. 2018. Vol. 43, no. 3. P. 37–42. URL: https://doi.org/10.31649/1999-9941-2018-43-3-37-42 (date of access: 17.11.2024).

9. Kiewkanya M., Surak S. Constructing C++ software size estimation model from class diagram. *2016 13th International Joint Conference on Computer Science and*

Software Engineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016. 2016. URL: https://doi.org/10.1109/jcsse.2016.7748880 (date of access: 17.11.2024).

10. Tan H.B.K., Zhao Y., Zhang H. Estimating LOC for information systems from their conceptual data models. *Proceeding of the 28th international conference, Shanghai*, China, 20–28 May 2006. New York, New York, USA, 2006. URL: https://doi.org/10.1145/1134285.1134331 (date of access: 17.11.2024).

11. Tan H.B.K., Zhao Y., Zhang H. Conceptual data model-based software size estimation for information systems. *ACM Transactions on Software Engineering and Methodology.* 2009. Vol. 19, no. 2. P. 1–37. URL: https://doi.org/10.1145/1571629.1571630 (date of access: 17.11.2024).

12. Prykhodko S.B., Prykhodko N.V., Makarova L.M. Estimating the Software Size of Open-Source PHP-Based Systems Using Non-Linear Regression Analysis. *Proceedings of International Conference "Advanced Computer Information Technologies" (ACIT-2018)*: CEUR Workshop Proceedings, Ceske Budejovice, CZECH REPUBLIC. Ceske Budejovice, 2019. P. 199–202.

13. 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.

14. Prykhodko S.B., Shutko I.S., Prykhodko A.S. A nonlinear regression model to estimate the size of web apps created using the CakePHP framework. *Radio Electronics, Computer Science, Control.* 2022. No. 4. P. 129–139. URL: https://doi.org/10.15588/1607-3274-2021-4-12 (date of access: 17.11.2024).

15. Prykhodko S., Shutko I., Prykhodko A. Early LOC Estimation of Web Apps Created Using Yii Framework by Nonlinear Regression Models. *WSEAS transactions on computers*. 2021. Vol. 20. P. 321–328. URL: https://doi.org/10.37394/23205.2021.20.35 (date of access: 17.11.2024).

16. Prykhodko S., Shutko I., Prykhodko A. Early size estimation of web apps created using CodeIgniter framework by nonlinear regression models. *Radioelectronic*

and computer systems. 2022. No. 3. P. 84–94. URL: https://doi.org/10.32620/reks.2022.3.06 (date of access: 17.11.2024).

17. 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.

18. Тубольцев Р.О., Пухалевич А.В. Нелінійна регресійна модель для оцінювання розміру e-commerce пакетів, розроблених мовою PHP. *Матеріали V Всеукраїнської науково-практичної інтернет конференції "Інформаційні технології: моделі, алгоритми, системи (ITMAS – 2024)" (30-31 жовтня 2024 р., м. Миколаїв)*. Миколаїв: НУК імені адмірала Макарова, 2024. С. 71-72.

19. 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.

20. Prykhodko N.V., Prykhodko S.B. Constructing the Nonlinear Regression Models on the Basis of Multivariate Normalizing Transformations. *Èlektronnoe modelirovanie*. 2018. Vol. 40, no. 6. P. 101–110. URL: https://doi.org/10.15407/emodel.40.06.101 (date of access: 17.11.2024).

21. Prykhodko S., Prykhodko N. Mathematical Modeling of Non-Gaussian Dependent Random Variables by Nonlinear Regression Models Based on the Multivariate Normalizing Transformations. *Advances in Intelligent Systems and Computing*. Cham, 2020. P. 166–174. URL: https://doi.org/10.1007/978-3-030-58124-4_16 (date of access: 17.11.2024).

22. Mardia K.V. Measures of multivariate skewness and kurtosis with applications. *Biometrika*. 1970. Vol. 57, no. 3. P. 519–530. URL: https://doi.org/10.1093/biomet/57.3.519 (date of access: 17.11.2024).

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

24. Prykhodko S.B., Prykhodko N.V., Makarova L.M., Kudin O.O., Smykodub T.G., Prykhodko A.S. Detecting bivariate outliers on the basis of normalizing

transformations for non-Gaussian data. *Advanced Information Systems and Technologies: proceedings of the V international scientific conference.* (Sumy, May 17-19 2017). Edited by S. I. Protsenko, V. V. Shendryk. Sumy: Sumy State University, 2017. P. 95-97. URL: http://essuir.sumdu.edu.ua/handle/123456789/55754 (date of access: 17.11.2024).

25. Prykhodko S., Prykhodko N., Makarova L., Pukhalevych A. Application of the Squared Mahalanobis Distance for Detecting Outliers in Multivariate Non-Gaussian Data. *Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET).* Lviv-Slavske, Ukraine, February 20–24, 2018. P. 962-965. URL: https://doi.org/10.1109/tcset.2018.8336353 (date of access: 17.11.2024).

26. Gunst R.F., Mason R.L. Regression Analysis and Its Application. CRC Press, 2018. URL: https://doi.org/10.1201/9780203741054 (date of access: 07.12.2024).

27. Foss T. A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering.* 2003. Vol. 29, no. 11. P. 985–995. URL: https://doi.org/10.1109/tse.2003.1245300 (date of access: 17.11.2024).

28. Prykhodko S., Prykhodko N. Mathematical Modeling of Non-Gaussian Dependent Random Variables by Nonlinear Regression Models Based on the Multivariate Normalizing Transformations. *Advances in Intelligent Systems and Computing*. Cham, 2020. P. 166–174. URL: https://doi.org/10.1007/978-3-030-58124-4_16 (date of access: 17.11.2024).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Найменування програми: «E-commerce size».

1 Підстави для розробки

Підставою для розробки є наказ на затвердження тем кваліфікаційних робіт магістрів.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням програми є надання користувачу можливості оцінити розмір модулів електронної комерції на мові PHP.

2.2 Експлуатаційне призначення

Програма призначена для експлуатації на персональних комп’ютерах користувачів для оцінювання розміру модулів електронної комерції на мові PHP.

3 Вимоги до програми або програмного продукту

3.1 Вимоги для функціональних характеристик

3.1.1 Вимоги до переліку виконуваних функцій

Розроблена програма має надавати можливість виконувати наступні функції:

– Оцінювання розміру розміру модулів електронної комерції на мові PHP у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру модулів електронної комерції на мові PHP.

– Оцінювання інтервалу передбачення значення розміру модулів електронної комерції на мові PHP.

3.1.2 Вимоги для організації вхідних та вихідних даних

Введення вхідних даних відбувається за рахунок заповнення полів графічного інтерфейсу користувача.

Результати виводяться у відповідні поля графічного інтерфейсу користувача.

Вхідними даними для побудови нелінійної регресійної моделі є набір метрик з проектів модулів електронної комерції на мові PHP.

Вхідними даними для проведення оцінювання розміру e-commerce модулів є такі метрики програмного забезпечення як кількість класів (Classes), кількість методів на кожен клас (Methods), глибина дерева наслідування (DIT). Також для виконання оцінювання необхідно вказати довірчу ймовірність.

Кількість класів e-commerce модулю – це ціле число, яке більше за одиницю.

Середня кількість методів на кожен клас, глибина дерева наслідування та довірча ймовірність, фактичний розмір – дійсні додатні числа. Довірча ймовірність приймає значення не більші за одиницю.

Вихідними даними є отримана оцінка розміру модуля електронної комерції у тисячах рядків коду, оцінки довірчого інтервалу та інтервалу передбачення, які буде відображено у відповідних полях графічного інтерфейсу програми.

3.2 Вимоги до надійності

Надійне (стійке) функціонування програмного забезпечення повинно бути забезпечено виконанням сукупності організаційно-технічних заходів, виконанням валідації вхідних даних згідно вимог та відображення повідомлень про виявлення некоректних даних з закликом ввести коректні.

3.3 Умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні задовольняти вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.4 Вимоги до технічної та програмної сумісності

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітний процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

3.5 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм не висуваються.

3.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки не висуваються.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4 Вимоги до програмної документації

Склад програмної документації має включати в себе:

1) Технічне завдання;
2) Програму та методику випробувань;
3) Інструкцію користувача;
4) Опис програмного забезпечення.
5) Текст програми

5 Техніко-економічні показники

Економічна ефективність від впровадження програмного забезпечення була розрахована у розділі 4. Виходячи з одержаних результатів, впровадження даного програмного забезпечення є доцільним, приблизний термін окупності – 4 місяці.

6 Стадії та етапи розробки програмного забезпечення

Стадії та етапи виконання розробки програмного забезпечення можна представити наступним чином:

Таблиця А.1 – Стадії та етапи розробки

Стадії розробки	Назва етапу	Термін виконання
Технічне завдання	Розробка технічного завдання	27.09.2024
	Затвердження технічного завдання	10.10.2024
Ескізний проект	Розробка ескізного проекту	22.10.2024
	Затвердження ескізного проекту	29.10.2024
Технічний проект	Розробка технічного проекту	16.11.2024
	Затвердження технічного проекту	27.11.2024
Робочий проект	Розробка програми	26.11.2024
	Розробка програмної документації	01.12.2024
	Випробування програми	05.12.2024

7 Порядок контролю та прийомки

Приймально-здавальні роботи мають проводитись під наглядом Замовника або уповноваженої ним особи.

Приймально-здавальні випробування повинні проводитись згідно документу «Програма та методика випробувань», який узгоджується між стороною розробника та стороною замовника.

Хід проведення приймально-здавальних робіт Замовник та Виконавець документують в «Протокол проведення випробувань».

На основі Протоколу проведення випробувань Виконавець та Замовник підписують «Акт прийому-здачі програми у експлуатацію».

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом дослідження є програмне забезпечення для оцінювання розміру модулів електронної комерції на мові PHP.

2 Мета випробувань

Мета випробувань – перевірка працездатності програмного застосунку, перевірка відповідності фактичних характеристик програмного застосунку тим вимогам, які було викладено у технічному завданні.

3 Вимоги до застосунку

Під час виконання випробувань перевіряються функціональні характеристики (вимоги), що були викладені в технічному завданні.

Розроблена програма має надавати можливість виконувати наступні функції:

– Оцінювання розміру модулів електронної комерції на мові PHP у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру модулів електронної комерції на мові PHP.

– Оцінювання інтервалу передбачення значення розміру модулів електронної комерції на мові PHP.

4 Вимоги до програмної документації

Програмна документація включає в себе наступні документи:

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

Склад технічних засобів, що використовувались під час випробувань:

Acer Aspire 3 A315-59-51ST (NX.K6SEU.00M):

– Десяти ядерний Intel Core i5-1235U (1.3 – 4.4 ГГц).

– Оперативна пам’ять: 16 ГБ, DDR4.

– Накопичувач даних: SSD 512 ГБ.

– Графіка: Intel Iris Xe Graphics.

– Екран: 15.6" IPS Full HD (1920x1080), 60 Гц.

5.2 Програмні засоби, що використовувались під час випробувань

Операційна система Windows 10 Pro 21H2 204.2125

5.3 Порядок проведення випробувань

Порядок проведення випробувань – перевірка відповідності функціональних характеристик програмного застосунку, перевірка вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

За відсутності конкретних вимог до оформлення та подання програмної документації, розроблені програмні документи перевіряються представником замовника візуально. Уповноважена особа перевіряє відповідність розроблених програмних документів з тими, що були вказані у пункті «Склад програмної документації, що пропонується на випробування».

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» технічного завдання.

Перевірку можна вважати успішно завершеною у випадку відповідності виконуваних функції технічному завданню.

Порядок випробувань:

– Перевірка роботи застосунку на моніторах з різною розподільною здатністю.

– Перевірка відображення шрифтів.

– Перевірка кожної форми графічного інтерфейсу на можливість введення даних.

– Перевірка кожної форми графічного інтерфейсу на введення невалідних даних.

– Перевірка можливості виконання оцінювання розміру, визначення довірчого інтервалу та інтервалу передбачення для модулів електронної комерції на мові PHP.

7 Результати випробувань

За результатами випробувань було підтверджено відповідність функціональних можливостей висунутим вимогам до програмного забезпечення.

8 Відомості про відмови, збої та аварійні ситуації

Під час проведення випробувань відмов, збоїв та аварійних ситуацій помічено не було

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

Коригування параметрів об’єкта випробувань та технічної документації в процесі виконання випробувань не проводилось

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА

Вступ

Інструкція користувача призначена для ознайомлення користувача з правилами та порядком виконання операцій, які забезпечуються роботою програмного забезпечення.

Ця програма призначена для оцінювання розміру та визначення інтервальних оцінок розміру у тисячах строк коду модулів електронної комерції на мові PHP.. Програма є вільним програмним забезпеченням з графічним інтерфейсом користувача з можливостю вводити дані та переглядати результати.

Загальні відомості про програму

Найменування програми: «E-commerce size».

Використані мови програмування

Програма написана на мові Python, для розробки графічного інтерфейсу користувача було використано сумісну з Python бібліотеку Qt.

Призначення програми

Програма призначена для оцінювання розміру (в тисячах рядків коду) модулів електронної комерції на мові PHP на основі таких метрик як кількість класів (Classes, Number of Classes), середня кількість методів на клас (MBC, Methods by Class), глибина дерева наслідування (DIT, Depth of Inheritance Tree).

Функціональні можливості програми

– Оцінювання розміру модулів електронної комерції на мові PHP у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру модулів електронної комерції на мові PHP.

– Оцінювання інтервалу передбачення значення розміру модулів електронної комерції на мові PHP.

Обмеження області застосування програми

Область застосування програми обмежена галуззю розробки програмного

забезпечення.

Умови, необхідні для використання програми

Спеціальні умови для використання програми відсутні

Вимоги до технічної сумісності та програмного середовища

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітній процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

На рисунку В.1 показано стартове вікно програми «E-commerce size». Вікно містить кілька полів для введення даних:

– «Data File» – вибір файлу для побудови регресійної моделі.

– «Classes (X1)» – введення кількості класів застосунку (невід’ємне число >1).

– «Methods (X2)» – введення середньої кількості методів на клас (невід’ємне число >1).

– «DIT (X3)» – середня глибина дерева наслідування (невід’ємне число).

– «Confidence Level» — рівень довірчої ймовірності (від 0 до 1 включно).

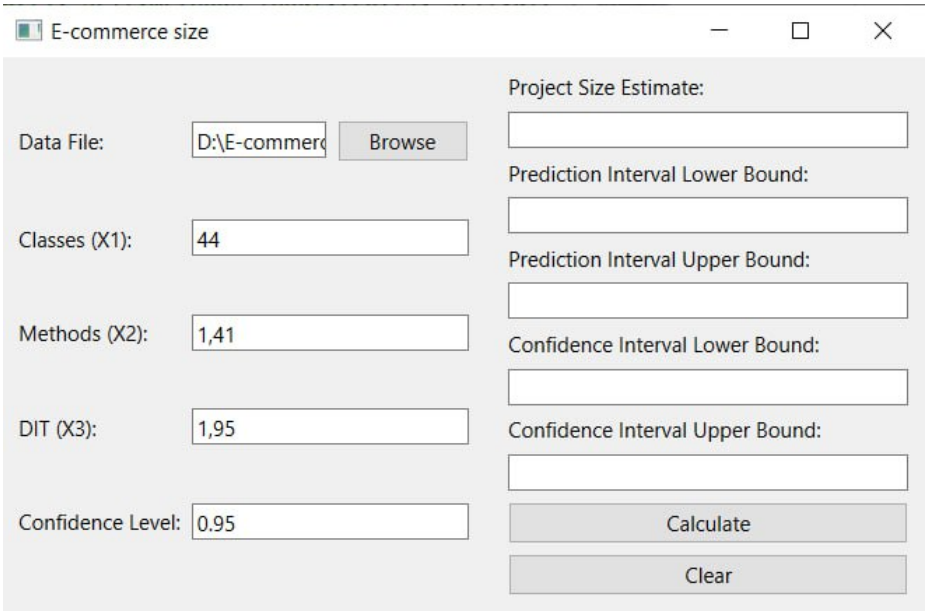

Рисунок В.1 – Вікно програми після виконання оцінювання

Після введення даних і запуску обчислень результати, включаючи розмір застосунку (у тисячах строк коду), довірчі інтервали та прогнози, відображаються в нижній частині екрана. Результат оцінювання наведено на рисунку В.2.

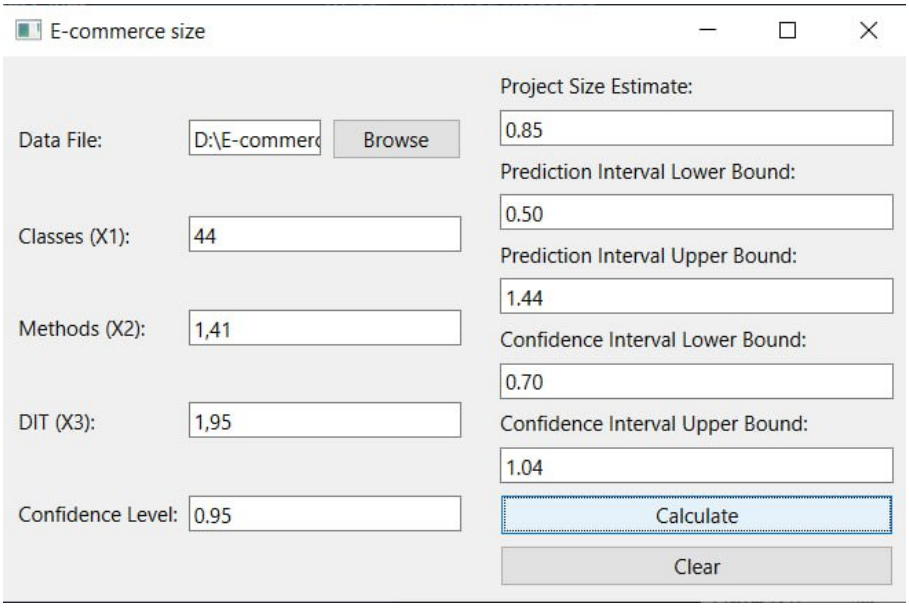

Рисунок В.2 – Вікно програми після виконання оцінювання

ДОДАТОК Г – ТЕКСТ ПРОГРАМИ

```
import numpy as np
import pandas as pd
import math
from scipy.stats import norm
from scipy.stats import chi2
from scipy.stats import t
from scipy.stats import chisquare
from scipy.stats import normaltest
from scipy.stats import shapiro
from scipy.stats import chi2_contingency
from statistics import NormalDist
from sklearn import linear_model
import statsmodels.api as sm

class DataProcessor:
    def __init__(self, input_data):
        self.dataset = input_data.copy()
        self.num_columns = len(self.dataset.columns)
        self.reduced_columns = self.num_columns - 1

    def compute_mahalanobis_distance(self, data_matrix):
        adjusted_data = data_matrix - np.mean(data_matrix, axis=0)

        covariance_matrix = np.cov(data_matrix.values.T, bias=True)

        inv_cov_matrix = np.linalg.inv(covariance_matrix)

        dot_product = np.dot(adjusted_data, inv_cov_matrix)

        mahalanobis_distances = np.dot(dot_product, adjusted_data.T)
        return mahalanobis_distances.diagonal()

    def apply_transformation(self, transform_func, dataset=None):
        if dataset is None:
            dataset = self.dataset.copy()

        column_names = dataset.columns

        for i, column in enumerate(column_names):
            new_column = 'Transformed_' + column
            dataset[new_column] = dataset.iloc[:, i].map(lambda x: transform_func(x))

        return dataset

    def identify_outliers(self, dataset):
```

```
        num_features = self.num_columns
        while True:
            dataset['mahalanobis_distance'] = self.compute_mahalanobis_distance(
                data_matrix=dataset.iloc[:, num_features:num_features+num_features]
            )
            chi_square_threshold = chi2.ppf(1 - 0.05, self.num_columns)
            max_index = dataset['mahalanobis_distance'].idxmax()
            max_distance = dataset['mahalanobis_distance'].max()

            if max_distance >= chi_square_threshold:
                dataset = dataset.drop(index=max_index)
            else:
                break

        return dataset

    def calculate_polynomial_regression(self, dataset, coefficients,
intercept_value):
        return (dataset ** coefficients).prod(axis=1) * 10 ** intercept_value

    def compute_project_metrics(self, metric_data, confidence=0.05):
        metric_data = self.apply_transformation(transform_func=math.log10,
dataset=metric_data)

        adjusted_data = metric_data.iloc[:, self.reduced_columns:] - self.mean_values

        metric_data['regression_prediction'] = self.calculate_polynomial_regression(
            dataset=metric_data.iloc[:, 0:self.reduced_columns],
            coefficients=self.linear_model.coef_,
            intercept_value=self.linear_model.intercept_
        )

        metric_data['scalar_values'] = np.dot(np.dot(adjusted_data,
self.inverse_covariance), adjusted_data.T)

        t_quantile = t.isf((1 - confidence) / 2, self.final_sample_size -
self.reduced_columns - 1)

        confidence_delta = t_quantile * self.standard_error * math.sqrt(1 /
self.final_sample_size + float(metric_data['scalar_values'][0]))
        prediction_delta = t_quantile * self.standard_error * math.sqrt(1 + (1 /
self.final_sample_size) + float(metric_data['scalar_values'][0]))

        metric_data['confidence_lower'] = 10 **
(math.log10(metric_data['regression_prediction'][0]) - confidence_delta)
        metric_data['confidence_upper'] = 10 **
(math.log10(metric_data['regression_prediction'][0]) + confidence_delta)
        metric_data['prediction_lower'] = 10 **
(math.log10(metric_data['regression_prediction'][0]) - prediction_delta)
```

```
        metric_data['prediction_upper'] = 10 **
(math.log10(metric_data['regression_prediction'][0]) + prediction_delta)

        return metric_data

    def compute_scalar_values(self, dataset=None):
        column_names = dataset.columns

        num_features = len(column_names)

        self.mean_values = np.mean(dataset, axis=0)
        centered_data = dataset - np.mean(dataset, axis=0)

        for i in range(num_features):
            new_column = column_names[i] + '_squared'
            centered_data[new_column] = centered_data.iloc[:, i].map(lambda x: x * x)

        matrix_result = np.zeros((num_features, num_features))

        for i in range(num_features):
            for j in range(num_features):
                matrix_result[i][j] = (centered_data.iloc[:, i] *
centered_data.iloc[:, j]).sum()

        relevant_data = centered_data.iloc[:, 0:self.reduced_columns]

        result_array = np.array([])

        inverse_matrix = np.linalg.inv(matrix_result)
        self.inverse_covariance = inverse_matrix

        for _, row in relevant_data.iterrows():
            temporary_result = np.dot(np.dot(row, inverse_matrix), row.T)
            result_array = np.append(result_array, temporary_result)

        return pd.DataFrame(result_array)

    def compute_chi_square_statistic(self, series, num_intervals=6):
        count = len(series)

        avg = np.mean(series)
        deviation = np.std(series, ddof=1)

        minimum = series.min()
        maximum = series.max()
        interval_width = (maximum - minimum) / num_intervals

        lower_bound = minimum
        upper_bound = lower_bound + interval_width
```

```
        chi_square_sum = 0
        for i in range(num_intervals):
            if i == num_intervals - 1:
                interval_values = series[(series <= maximum) & (series >=
lower_bound)]
            else:
                interval_values = series[(series < upper_bound) & (series >=
lower_bound)]

            prob = norm(avg, deviation).cdf(upper_bound) - norm(avg,
deviation).cdf(lower_bound)

            chi_square_sum += (len(interval_values) - count * prob) ** 2 / (count *
prob)

            lower_bound = upper_bound
            upper_bound += interval_width

        return chi_square_sum


    def calculate_standard_error(self, dataset):
        total_samples = len(dataset)

        sum_squared_residuals =
dataset[['linear_residuals']].pow(2).sum()['linear_residuals']

        return math.sqrt(1.0 / (total_samples - self.reduced_columns - 1) *
sum_squared_residuals)

    def perform_nonlinear_regression(self):
        dataset = self.dataset

        while True:
            normalized_data = self.apply_transformation(transform_func=math.log10,
dataset=dataset.iloc[:, :self.num_columns])

            normalized_data = self.identify_outliers(dataset=normalized_data)

            normalized_data = normalized_data.reset_index().drop(labels='index',
axis=1)

            predictors = normalized_data.iloc[:,
self.num_columns+1:2*self.num_columns]
            target = normalized_data.iloc[:, self.num_columns]

            self.linear_model = linear_model.LinearRegression()
            self.linear_model.fit(predictors, target)

            model_results = sm.OLS(target, predictors).fit()
```

```
            predictions = self.linear_model.predict(predictors)

            normalized_data['linear_predictions'] = pd.DataFrame(predictions)
            normalized_data['linear_residuals'] =
normalized_data['linear_predictions'] - normalized_data['Transformed_Y']

            chi_critical_value = chi2.ppf(1 - 0.05, 3)
            chi_square_value =
self.compute_chi_square_statistic(series=normalized_data['linear_residuals'])

            normalized_data['nonlinear_predictions'] =
self.calculate_polynomial_regression(
                dataset=normalized_data.iloc[:, 1:self.num_columns],
                coefficients=self.linear_model.coef_,
                intercept_value=self.linear_model.intercept_
            )

            normalized_data['scalar_values'] =
self.compute_scalar_values(dataset=normalized_data.iloc[:,
self.num_columns+1:self.num_columns*2])

            self.final_sample_size = len(normalized_data)

            self.standard_error =
self.calculate_standard_error(dataset=normalized_data)

            t_quantile = t.isf(0.025, self.final_sample_size - self.reduced_columns -
1)

            normalized_data['confidence_delta'] =
normalized_data[['scalar_values']].map(lambda x: t_quantile * self.standard_error *
math.sqrt(1 / self.final_sample_size + x))
            normalized_data['prediction_delta'] =
normalized_data[['scalar_values']].map(lambda x: t_quantile * self.standard_error *
math.sqrt(1 + (1 / self.final_sample_size) + x))

            normalized_data['confidence_lower'] = 10 **
(normalized_data['linear_predictions'] - normalized_data['confidence_delta'])
            normalized_data['confidence_upper'] = 10 **
(normalized_data['linear_predictions'] + normalized_data['confidence_delta'])
            normalized_data['prediction_lower'] = 10 **
(normalized_data['linear_predictions'] - normalized_data['prediction_delta'])
            normalized_data['prediction_upper'] = 10 **
(normalized_data['linear_predictions'] + normalized_data['prediction_delta'])

            outlier_data = normalized_data[(normalized_data.Y <=
normalized_data.prediction_lower) | (normalized_data.Y >=
normalized_data.prediction_upper)]

            if chi_critical_value > chi_square_value and outlier_data.empty:
```

```
                break
            elif chi_critical_value < chi_square_value:
                if abs(normalized_data['linear_residuals'].max()) >
abs(normalized_data['linear_residuals'].min()):
                    residual_to_remove = normalized_data['linear_residuals'].idxmax()
                else:
                    residual_to_remove = normalized_data['linear_residuals'].idxmin()

                normalized_data = normalized_data.drop(index=residual_to_remove)
                dataset = normalized_data
            elif not outlier_data.empty:
                normalized_data = normalized_data.drop(outlier_data.index)
                dataset = normalized_data


import pandas as pd
from DataProcessor import DataProcessor
import sys
from PyQt6.QtWidgets import (
    QApplication, QWidget, QFileDialog, QPushButton, QLineEdit, QLabel, QVBoxLayout,
QMainWindow, QGridLayout, QScrollArea, QSplitter
)
from PyQt6.QtGui import QDoubleValidator, QIntValidator
from pathlib import Path

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('E-commerce size')

        main_splitter = QSplitter()
        self.setCentralWidget(main_splitter)

        input_widget = QWidget()
        input_layout = QGridLayout(input_widget)

        self.file_path = QLineEdit(self)
        browse_button = QPushButton('Browse')
        browse_button.clicked.connect(self.file_dialog)

        self.classes_input = QLineEdit(self)
        self.classes_input.setValidator(QIntValidator(self))
        self.methods_input = QLineEdit(self)
        self.methods_input.setValidator(QDoubleValidator(self))
        self.dit_input = QLineEdit(self)
        self.dit_input.setValidator(QDoubleValidator(self))
        self.confidence_input = QLineEdit("0.95", self)
        self.confidence_input.setValidator(QDoubleValidator(0.0, 1.0, 2,
parent=self))
```

```
input_layout.addWidget(QLabel("Data File:"), 0, 0)
input_layout.addWidget(self.file_path, 0, 1)
input_layout.addWidget(browse_button, 0, 2)

input_layout.addWidget(QLabel("Classes (X1):"), 1, 0)
input_layout.addWidget(self.classes_input, 1, 1, 1, 2)

input_layout.addWidget(QLabel("Methods (X2):"), 2, 0)
input_layout.addWidget(self.methods_input, 2, 1, 1, 2)

input_layout.addWidget(QLabel("DIT (X3):"), 3, 0)
input_layout.addWidget(self.dit_input, 3, 1, 1, 2)

input_layout.addWidget(QLabel("Confidence Level:"), 4, 0)
input_layout.addWidget(self.confidence_input, 4, 1, 1, 2)

main_splitter.addWidget(input_widget)

output_widget = QWidget()
output_layout = QVBoxLayout(output_widget)

self.project_size_output = QLineEdit(self)
self.project_size_output.setReadOnly(True)

self.pred_lower_output = QLineEdit(self)
self.pred_lower_output.setReadOnly(True)

self.pred_upper_output = QLineEdit(self)
self.pred_upper_output.setReadOnly(True)

self.conf_lower_output = QLineEdit(self)
self.conf_lower_output.setReadOnly(True)

self.conf_upper_output = QLineEdit(self)
self.conf_upper_output.setReadOnly(True)

output_layout.addWidget(QLabel("Project Size Estimate:"))
output_layout.addWidget(self.project_size_output)

output_layout.addWidget(QLabel("Prediction Interval Lower Bound:"))
output_layout.addWidget(self.pred_lower_output)

output_layout.addWidget(QLabel("Prediction Interval Upper Bound:"))
output_layout.addWidget(self.pred_upper_output)

output_layout.addWidget(QLabel("Confidence Interval Lower Bound:"))
output_layout.addWidget(self.conf_lower_output)

output_layout.addWidget(QLabel("Confidence Interval Upper Bound:"))
output_layout.addWidget(self.conf_upper_output)
```

```
        actions_layout = QVBoxLayout()

        self.calculate_button = QPushButton('Calculate')
        self.calculate_button.clicked.connect(self.calculate)
        self.clear_button = QPushButton('Clear')
        self.clear_button.clicked.connect(self.clear_fields)

        actions_layout.addWidget(self.calculate_button)
        actions_layout.addWidget(self.clear_button)

        output_layout.addLayout(actions_layout)
        main_splitter.addWidget(output_widget)

        main_splitter.setStretchFactor(0, 2)
        main_splitter.setStretchFactor(1, 3)

    def file_dialog(self):
        filename, _ = QFileDialog.getOpenFileName(self, "Select a File", "", "Metrics
files (*.xlsx *.csv *.txt)")
        if filename:
            self.file_path.setText(str(Path(filename)))

    def clear_fields(self):
        self.file_path.clear()
        self.classes_input.clear()
        self.methods_input.clear()
        self.dit_input.clear()
        self.confidence_input.setText("0.95")
        self.project_size_output.clear()
        self.pred_lower_output.clear()
        self.pred_upper_output.clear()
        self.conf_lower_output.clear()
        self.conf_upper_output.clear()

    def calculate(self):
        file_path = self.file_path.text()
        if not all([self.classes_input.text(), self.methods_input.text(),
self.dit_input.text(), self.confidence_input.text(), file_path]):
            return

        classes = float(self.classes_input.text().replace(',', '.'))
        methods = float(self.methods_input.text().replace(',', '.'))
        dit = float(self.dit_input.text().replace(',', '.'))
        confidence_level = float(self.confidence_input.text().replace(',', '.'))

        data = pd.read_excel(file_path)
        dataProcessor = DataProcessor(data)
        dataProcessor.perform_nonlinear_regression()
```

```
        result = dataProcessor.compute_project_metrics(
            metric_data=pd.DataFrame({'X1': [classes], 'X2': [methods], 'X3':
[dit]}),
            confidence=confidence_level
        )

        self.project_size_output.setText('%.2f' % result['regression_prediction'][0])
        self.pred_lower_output.setText('%.2f' % result['prediction_lower'][0])
        self.pred_upper_output.setText('%.2f' % result['prediction_upper'][0])
        self.conf_lower_output.setText('%.2f' % result['confidence_lower'][0])
        self.conf_upper_output.setText('%.2f' % result['confidence_upper'][0])

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

ДОДАТОК Д – ОПИС ПРОГРАМИ

1 Загальні відомості

Найменування: «E-commerce size».

1.2 Програмне забезпечення, необхідне для функціонування програми

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

1.3 Мови програмування

Програмне забезпечення для оцінювання розміру модулів електронної комерції на мові PHP., розроблене з використанням мови програмування Python та використанням бібліотек Pandas, NumPy та SciPy для виконання математичних обчислень. Для створення графічного інтерфейсу користувача було використано сумісну з Python бібліотеку Qt.

2 Функціональне призначення

Функціональним призначенням програми є надання користувачу можливості оцінити розмір модулів електронної комерції на мові PHP

Функції програмного застосунку, які доступні користувачу:

– Оцінювання розміру модулів електронної комерції на мові PHP у тисячах строк коду.

– Оцінювання довірчого інтервалу значення розміру модулів електронної комерції на мові PHP.

– Оцінювання інтервалу передбачення значення розміру модулів електронної комерції на мові PHP.

2.1 Функціональні обмеження

Програмне забезпечення не вимагає використання зовнішніх ресурсів, тому функціональні обмеження відсутні.

3 Технічні засоби, що використовуються

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні: 64-бітній процесор на архітектурі x86 або ARM, оперативна пам’ять 2Гб (1 Гб мінімум), простір на жорсткому диску 100 Мб мінімум.

4 Виклик та завантаження

Для запуску та подальшої роботи програмного забезпечення необхідна операційна система Windows 7/8/10/11, Ubuntu 17/18/19/20.

Запуск програмного застосунку відбувається за рахунок натискання на ярлик програмного застосунку.

5 Вхідні дані

Вхідними даними для побудови нелінійної регресійної моделі є файл з метриками з розширенням .txt/.csv/.xlsx в якому в першій колонці знаходиться залежна величина – кількість рядків коду у тисячах, а у другому-четвертому кількість класів, кількість методів на кожен клас та глибина дерева наслідування відповідно.

Вхідними даними для оцінювання модулів електронної комерції на мові PHP є кількість класів, середня кількість методів на кожен класів, глибина дерева наслідування, довірча ймовірність.

Кількість класів e-commerce модуля – це ціле число, яке більше за одиницю.

Значення середньої кількості методів, глибини дерева наслідування e-commerce модуля застосунку мають бути дійсними додатними числами.

Довірча ймовірність вказує на відсотки та є дійсним числом від нуля до одиниці включно.

6 Вихідні дані

Вихідними даними є отримані оцінки розміру у тисячах рядків коду, оцінки довірчого інтервалу та інтервалу передбачення. Отримані значення буде відображено у відповідних полях графічного інтерфейсу програми.