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

_____
«___» _____ 2021 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, та розробка програмного забезпечення для її реалізації

Виконала: студентка 6151м групи
Кузнецова А.С.
(підпис, ПІБ)

Керівник роботи:
к.т.н., доцент
(посада, науковий ступень вчене звання)
Устенко І.В.
(підпис, ПІБ)

Миколаїв – 2021 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)
«___» _____ 2021 р.

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

Студенту _____Кузнецовій Анастасії Сергіївні_____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, та розробка програмного забезпечення для її реалізації_____

Керівник роботи Устенко Ірина Валеріївна к.т.н., доцент_____

Затверджені наказом ректора № 1239уч від «13» _____10_____ 2021 р.

2. Термін подання роботи: 06.12.2021 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 Тема кваліфікаційної роботи, Актуальність роботи, Мета та завдання дослідження, Об'єкт та предмет дослідження, наукова новизна, Апробація результатів досліджень, Збір даних для побудови регресійної моделі, Розподіл значень випадкових величин, Лінійна регресійна модель, Зворотне перетворення, Нелінійна регресійна модель, Метрики якості регресійних моделей, Оцінка якості лінійної та нелінійної регресійних моделей, Постановка задачі на розробку програмного забезпечення, Діаграма варіантів використання, Діаграми діяльності основних варіантів використання, Діаграма класів, Інтерфейс програмного забезпечення, Результати кваліфікаційної роботи, Висновки._____

_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 13.10.2021 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	18.10.2021	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	19.10.2021	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	22.10.2021	НС*
4.	Підготовка розділу з проекту програмного забезпечення	16.11.2021	
5.	Підготовка організаційно-економічного розділу	19.11.2021	
6.	Підготовка розділу з охорони праці	22.11.2021	
7.	Підготовка розділу з охорони навколишнього середовища	24.11.2021	
8.	Підготовка розділу Висновки	25.11.2021	
9.	Оформлення списку використаних джерел та додатків	29.11.2021	
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.)	06.12.2021	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
11.	Підготовка презентації та доповіді	09.12.2021	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	10.12.2021	
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	20.12.2021	

* - за результатами наукового стажування (НС), яке було з 01.09.2021 до 10.10.2021 р.

Студент _____ Кузнецова А.С.
(підпис) (ПІБ)

Керівник роботи _____ Устенко І.В.
(підпис) (ПІБ)

РЕФЕРАТ

Кузнецова Анастасія Сергіївна

«Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, та розробка програмного забезпечення для її реалізації»

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2021 р.

Обсяг роботи: 112 стор., 22 табл., 21 рис., 20 використаних джерел, 5 додатків.

Актуальність теми роботи. Під час розробки мобільних застосунків необхідним є етап планування, а саме процес оцінювання розміру програмного забезпечення, який є першим кроком та основою для подальшого розрахунку трудомісткості та ресурсів, необхідних для реалізації проекту. Вагомий вплив на трудомісткість має розмір мобільних застосунків. Проте не існує моделей для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin. Таким чином, удосконалення регресійної моделі для оцінювання розміру мобільних застосунків на Kotlin є актуальним завданням.

Мета та завдання дослідження. Метою даної роботи є підвищення достовірності оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, та розробка програмного забезпечення для її реалізації. Завдання дослідження: провести аналіз існуючих методів та моделей для оцінювання розміру мобільних застосунків; обґрунтувати необхідність удосконалення математичної моделі; удосконалити нелінійну регресійну модель; розробити програмне забезпечення для оцінювання розміру мобільних застосунків на Kotlin.

Об’єкт дослідження: процес оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Предмет дослідження: нелінійна регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Методи дослідження. Для вирішення поставлених задач були застосовані методи теорії ймовірностей, математичної статистики та регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру мобільних застосунків, які створюються мовою Kotlin, що дозволяє підвищити достовірність оцінювання розміру мобільних застосунків у порівнянні з лінійними моделями.

Практичне значення одержаних результатів. Розроблено програмне забезпечення мовою програмування C# для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у роботі, були апробовані на IV Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні комп’ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2021 р.).

Ключові слова: НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ, НОРМАЛІЗУЮЧЕ ПЕРЕТВОРЕННЯ, ОЦІНЮВАННЯ РОЗМІРУ, МОБІЛЬНИЙ ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ KOTLIN.

ABSTRACT

Kuznetsova Anastasiia

«Regression model for estimating the size of mobile applications created using Kotlin and developing a software for its implementation»

Qualification work for gaining a master's degree in specialty 121 – "Software Engineering". Admiral Makarov National University of Shipbuilding. Mykolayiv, 2021.

The work contains 112 pages, 22 tables, 21 figures, 20 references, 5 appendices.

Relevance of the topic: Estimating of mobile application characteristics is an important step in the software project planning process. Most of these characteristics are based on estimating the size of mobile applications. Size estimation has the greatest impact on the complexity and cost of the software. But there are no universal methods or models that are optimal for estimating size of all types of mobile applications. Therefore, the task of improvement of nonlinear regression model for estimating the size of mobile applications created using the Kotlin programming language is relevant.

Purpose and tasks of the research: The purpose of this qualification work is to increase the reliability of estimating the size of mobile applications created using the Kotlin programming language and develop a program for its implementation.

Tasks include: analysis of existing methods and models for estimating the size of mobile applications, improving the nonlinear regression model for estimating the size of mobile applications created using the Kotlin programming language, software development for its implementation.

Object of research: the process of estimating the size of mobile applications created using the Kotlin programming language.

Subject of research: nonlinear regression model for estimating the size of mobile applications created using the Kotlin programming language.

Methods of research. Methods of probability theory, mathematical statistics and regression analysis were used to solve the problems.

Scientific novelty of the obtained results: improved regression model for estimating the size of mobile applications created using the Kotlin programming language, which allowed to obtain a more accurate and reliable estimate of the size.

The practical value of the results. Software for estimating the size of mobile applications created using the Kotlin programming language.

Approbation of study results. The main provisions and research results presented in the qualification work were tested at the IV All-Ukrainian scientific-practical Internet conference of students, graduate students and young scientists on «Modern computer systems and networks in management» (Kherson, November 30, 2021).

Keywords: NONLINEAR REGRESSION MODEL, NORMALIZING TRANSFORMATION, SIZE ESTIMATION, MOBILE APPLICATION, KOTLIN PROGRAMMING LANGUAGE.

ЗМІСТ

ВСТУП

Актуальність теми. Під час розробки мобільних застосунків необхідним є етап планування. Успішне завершення проекту значно залежить від цього етапу, тому важливим є процес оцінювання, який є першим кроком та основою для подальшого планування.

Зазвичай при розробці мобільних застосунків оцінюють трудомісткість та ресурси, які необхідні для реалізації проекту. Вагомий вплив на трудомісткість має розмір мобільних застосунків. Оцінювання розміру часто є основою для подальшого визначення тривалості розробки, трудомісткості, а також вартості виконаного проекту [1].

Існують різні методи оцінювання розміру програмних продуктів. Деякі з них є універсальними, інші створені для планування певних класів проектів. Серед найбільш поширених можна виділити наступні: експертна оцінка, регресійні моделі, метод функціональних точок, використання нейронних мереж, комбіновані методи [1]. Проте існуючі методи використовуються для інших класів проектів, потребують наявності експертних навичок, або є недостатньо точними. Наприклад серед регресійних моделей [2] існують математичні моделі для оцінювання розміру систем на Java [3], застосунків на PHP [4] та інших видів програмного забезпечення, але не існує методів та моделей для оцінювання розміру мобільних застосунків на Kotlin.

Отже, удосконалення регресійної моделі для оцінювання розміру мобільних застосунків на Kotlin та розробка програмного забезпечення для її реалізації є актуальним завданням. Удосконалена регресійна модель дозволить підвищити точність оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Мета і завдання дослідження. Метою даної роботи є підвищення достовірності оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Для досягнення поставленої мети потрібно вирішити наступні завдання:

- провести аналіз існуючих методів та моделей для оцінювання розміру мобільних застосунків;
- обґрунтувати необхідність удосконалення регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin;
- удосконалити регресійну модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin;
- розробити програмне забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin;

Об’єкт дослідження: процес оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Предмет дослідження: регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Методи дослідження. Для вирішення поставлених задач кваліфікаційної роботи були застосовані методи теорії ймовірностей, математичної статистики та регресійного аналізу.

Наукова новизна одержаних результатів: удосконалено нелінійну регресійну модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, з використанням нормалізуючого перетворення на основі натурального логарифму, яке дозволяє підвищити достовірність оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, у порівнянні з лінійними моделями.

Практичне значення одержаних результатів полягає в розробці програмного забезпечення мовою програмування C# для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, на основі удосконаленої регресійної моделі.

Особистий внесок здобувача. Дана робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто.

У роботі, що опублікована у співавторстві [5], здобувачеві належить удосконалення регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у звіті з наукового стажування, були апробовані на IV Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні комп'ютерні системи та мережі в управлінні» (м. Херсон, 30 листопада 2021 р.).

Публікації. Основні результати роботи викладено у 1 науковій праці – матеріалах конференції.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОБІЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN

1.1 Мова програмування Kotlin для створення мобільних застосунків

Kotlin – це об'єктно-орієнтована, кросплатформна, статично типізована, універсальна мова програмування, яка підтримує вивід типів та розробляється компанією JetBrains. Мова Kotlin працює поверх Java Virtual Machine та є повністю сумісною з Java, що дозволяє розробникам поступово переходити до її використання.

Мовою Kotlin можна створювати мобільні та веб-застосунки. Kotlin в основному націлений на використання віртуальної машини Java, але також компілюється у JavaScript.

Синтаксис Kotlin зазнав впливу наступних мов: JavaScript, Java, Scala, Groovy, C#, Python. Автори Kotlin намагались створити лаконічну, типово-безпечну та більш просту мову у порівнянні з Java та Scala. Наслідками спрощення стали більш швидка компіляція та краща підтримка середовища розробки.

Окрім об'єктно-орієнтованого підходу Kotlin також підтримує процедурний стиль з використанням функцій, що робить мову більш універсальною. Kotlin є open-source проектом, текст вихідного коду мови програмування знаходиться у відкритому доступі [6].

Kotlin є офіційно підтримуваною та пріоритетною мовою для розробки мобільних застосунків на Android [7]. Повна сумісність з Java дозволяє впроваджувати нові функції на Kotlin в існуючий мобільний застосунок без переписування програмного забезпечення повністю.

1.2 Метрики програмного забезпечення

Для оцінки програмного забезпечення використовують різні кількісні та якісні метрики. Зазвичай на практиці частіше використовуються кількісні метрики, оскільки вони легко піддаються підрахунку та статистичній обробці. Метрика програмного забезпечення дозволяє отримати числове значення для певної властивості програмного забезпечення.

Серед найпоширеніших метрик програмного забезпечення можна виділити наступні:

- кількість рядків коду;
- кількість класів;
- кількість модулів;
- кількість помилок на рядок коду;
- цикломатична складність;
- функціональні точки;
- покриття вимог;
- покриття коду тестуванням;
- метрики Холстеда [8].

Зазвичай ці метрики використовують як основу для розрахунку інших показників програмних проектів, наприклад трудомісткість, тривалість розробки, вартість. У процесі планування проектів важливим етапом є визначення розміру програмного забезпечення. Найбільш поширеною метрикою, яка визначає розмір мобільних застосунків, є кількість рядків коду. При використанні цієї метрики розмір програмного забезпечення розраховується як кількість рядків у тексті вихідного коду програмного забезпечення [9]. Кількість рядків коду можна визначити як кількість фізичних рядків з урахуванням коментарів, або як кількість логічних рядків, що залежить від мови програмування, якою створюється програмне забезпечення. Зазвичай кількість рядків вихідного коду мобільних застосунків є негаусівською випадковою величиною.

1.3 Аналіз існуючих методів і моделей для оцінювання розміру програмного забезпечення

Для оцінювання розміру мобільних застосунків існують різні методи та моделі. Деякі створені для конкретних видів програмних проектів, інші є більш універсальними. Серед найбільш широко використовуваних можна виділити наступні [10]:

- Експертне оцінювання. Оцінка розміру програмного проекту здійснюється на основі думок експертів, які мають досвід у розробці певного виду програмного забезпечення.

- Регресійні моделі. Розмір програмного проекту обчислюється на основі математичної моделі та заданих вхідних параметрів. При використанні регресійних моделей для оцінювання розміру програмного забезпечення будують лінійну або нелінійну модель [2]. Якщо випадкові величини, що входять до моделі, не підпорядковуються нормальному закону розподілу, то маємо нелінійну регресійну модель, яку можна побудувати за допомогою наступних методів: метод простого перебору, метод лінеаризуючих перетворень, метод нормалізуючих перетворень. Метод нормалізуючих перетворень полягає у застосуванні нормалізуючих перетворень, які дозволять перейти від негаусівських випадкових величин до величин з гаусівським законом розподілу. Для отриманих даних будується лінійне рівняння регресії, довірчий інтервал та інтервал передбачення. На основі зворотного перетворення будується нелінійна регресійна модель та відповідні інтервали [11]. Даний метод позбавлений недоліків, які мають методи лінеаризуючих перетворень та простого перебору.

- Метод функціональних точок. Метод дозволяє оцінити кількість функціональності на основі логічної моделі обсягу програмного продукту. Загальна кількість функціональних точок залежить від кількості певних елементарних процесів у програмному проекті [12].

- Нечітка логіка. Розмір програмного проекту оцінюється на основі розміру функцій.

- Використання нейронних мереж. Дозволяє оцінити розмір програмного проекту за аналогією з використанням штучного інтелекту та подальшого навчання нейронної мережі.

- Генетичні алгоритми. Покращений метод, який базується на використанні нейронних мереж.

- Комбіновані методи. Поєднують комбінацію двох або більше методів для отримання більш точної оцінки.

Сучасні методи та моделі продовжують розвиватись, але не існує універсального способу для оцінки розміру мобільних застосунків. Аналіз існуючих моделей та методів показав, що більшість існуючих методів потребують наявності експертів у сфері розробки мобільних застосунків або були створені для інших класів задач.

Отже, існуючі засоби та методи не дозволяють з необхідною точністю оцінювати розмір програмного забезпечення з урахуванням особливостей розробки мобільних застосунків мовою програмування Kotlin. Тому удосконалення моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, є актуальною задачею. Удосконалена модель дозволить підвищити достовірність прогнозування розміру мобільних застосунків.

1.4 Перевірка якості рівняння регресії для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin

Для оцінювання якості розробленої або вдосконаленої регресійної моделі використаємо наступні показники: коефіцієнт детермінації R^2, середня величина відносної похибки *MMRE*, рівень прогнозування PRED(*l*) [13, 14].

Коефіцієнт детермінації R^2 розрахуємо за наступною формулою:

$$R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right), \tag{1.1}$$

де y_i – емпіричне значення y;

$\hat{y}_i$ – розрахункове значення y;

$\bar{y}$ – середнє значення випадкової величини y.

Коефіцієнт детермінації R^2 змінюється від 0 до 1 та показує наскільки регресійна модель відповідає реальним даним. При $R^2 = 0$ можна зробити висновок про відсутність зв`язку між змінними регресійної моделі. Якщо значення $R^2 \geq 0{,}5$, то регресійну модель можна вважати прийнятною. Якщо $R^2 \geq 0{,}8$, то регресійна модель вважається результативною та достатньо ефективною. При $R^2 = 1$ лінія регресії вважається достовірною і точно відповідає спостереженням.

Значення середньої величини відносної похибки *MMRE* (Mean Magnitude Relative Error) розрахуємо за формулою (1.2).

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} MRE_i \,, \tag{1.2}$$

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

де величина відносної похибки *MRE* (Magnitude of Relative Errors) розраховується за формулою:

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{1.3}$$

де y_i – емпіричне значення випадкової величини y;

$\hat{y}_i$ – розрахункове значення за рівнянням регресії.

Також для визначення точності оцінки використаємо рівень прогнозування PRED(l), який розраховується за наступною формулою:

$$PRED(l) = \frac{k}{N}, \tag{1.4}$$

де k – кількість значень з $MRE \leq l$. Зазвичай l = 0,25. Прийнятне значення PRED(0,25) ≥ 0,75.

Отже, маємо ряд показників, які можна використовувати для перевірки якості регресійної моделі оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

1.5 Удосконалення регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin

Для підвищення точності оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, потрібно побудувати довірчий інтервал та інтервал прогнозування регресійної моделі. Але якщо вхідні дані не підпорядковуються нормальному закону розподілу, побудова відповідних інтервалів може бути ускладнена. У такому випадку необхідно виконати наступні дії [11]:

- За допомогою нормалізуючого перетворення отримати випадкові величини з нормальним законом розподілу [15].
- Побудувати лінійне рівняння регресії.
- Побудувати довірчий інтервал лінійного рівняння регресії за наступною формулою [16, 17]:

$$y = \hat{y} \pm t_{(\alpha/2, n-2)} \cdot S \cdot \sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}, \tag{1.5}$$

де $S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$,

$\hat{y}$ – значення y, яке розраховане за рівнянням регресії;

t – квантіль розподілу Стьюдента;

α – рівень значущості;

n – загальна кількість випадкових величин.

- Побудувати інтервал прогнозування наступним чином [16, 17]:

$$y = \hat{y} \pm t_{(\alpha/2,n-2)} \cdot S \cdot \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}, \quad (1.6)$$

де $S = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$,

$\hat{y}$ – значення y, яке розраховане за рівнянням регресії;

t – квантіль розподілу Стьюдента;

α – рівень значущості;

n – загальна кількість випадкових величин.

- За допомогою зворотного нормалізуючого перетворення отримати нелінійне рівняння регресії та відповідні інтервали для нелінійної регресійної моделі.

Таким чином отримаємо удосконалену регресійну модель оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, яку можна використовувати для прогнозування розміру мобільних застосунків.

1.6 Обґрунтування необхідності проведення досліджень з оцінювання розміру мобільних застосунків, що створюються мовою Kotlin

Відомо, що результат оцінювання розміру програмного забезпечення на основі регресійної моделі значно залежить від мови програмування, якою розробляється програмне забезпечення, та типу програмного проекту. Оскільки існуючі моделі розроблялись для інших видів проектів та не враховують особливості розробки мобільних застосунків на Kotlin, то удосконалення регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, є актуальною задачею. Удосконалена модель дозволить підвищити точність оцінювання розміру мобільних застосунків.

2 УДОСКОНАЛЕННЯ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОБІЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN

2.1 Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, із застосуванням нормалізуючих перетворень

Для побудови нелінійної регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, були зібрані вихідні дані з завершених проектів. У якості вихідних даних обрано кількість класів як незалежну змінну *X* та кількість рядків коду як залежну змінну *Y*.

Для перевірки відповідності вихідних даних нормальному закону розподілу використаємо критерій χ^2 [18]. Необхідно розрахувати χ^2 для емпіричних даних та порівняти з його критичним значенням для прийнятого рівня значущості α і числа ступенів свободи *n*. Якщо χ^2 менше ніж критичне значення χ^2, то з імовірністю 1-α приймається гіпотеза про те, що закон розподілу є нормальним.

Формула критерію має наступний вигляд [19]:

$$\chi^2 = \sum_{i=1}^{m} \frac{(n_i - np_i)^2}{np_i}, \tag{2.1}$$

де *m* – кількість підінтервалів, на яку розбивається вибірка *n*;

n_i – кількість даних, що потрапляють у *i*-й підінтервал;

p_i – теоретична ймовірність потрапляння випадкової величини в *i*-й підінтервал.

Якщо вихідні емпіричні дані не підпорядковуються нормальному закону розподілу, то для побудови нелінійної регресійної моделі спочатку необхідно виконати нормалізацію.

У даному випадку використовується нормалізуюче перетворення на основі натурального логарифму:

$$z = \ln x\,, \tag{2.2}$$

де z – випадкова величина з нормальним законом розподілу;

x – значення випадкової величини, яке нормалізується.

Зворотне перетворення до (2.2) для повернення до вихідних даних має наступний вигляд:

$$x = e^{z}\,. \tag{2.3}$$

В результаті застосування нормалізуючого перетворення (2.2) отримаємо значення випадкових величин z_x та z_y з нормальним законом розподілу. Для отриманих випадкових величин, що відповідають нормальному закону розподілу, можна побудувати лінійне рівняння регресії, яке має наступний вигляд:

$$z_y = b_1 z_x + b_0 + \varepsilon\,, \tag{2.4}$$

де b_0, b_1 – коефіцієнти лінійної регресії, що обчислюються методом найменших квадратів за наступною формулою:

$$b_1 = \frac{n\sum_{i=1}^{n} z_{xi} z_{yi} - \sum_{i=1}^{n} z_{xi} \cdot \sum_{i=1}^{n} z_{yi}}{n\sum_{i=1}^{n} z_{xi}^2 - (\sum_{i=1}^{n} z_{xi})^2},$$

$$b_0 = \frac{\sum_{i=1}^{n} z_{yi} - b_1 \sum_{i=1}^{n} z_{xi}}{n}. \tag{2.5}$$

Для отриманого лінійного рівняння регресії побудуємо довірчий інтервал за формулою (1.5) та інтервал прогнозування згідно з (1.6).

На основі лінійного рівняння регресії (2.4) за допомогою зворотного нормалізуючого перетворення (2.3) будуємо нелінійну регресію:

$$y = x^{b_1} \cdot e^{b_0 + \varepsilon}. \tag{2.6}$$

Довірчий інтервал та інтервал прогнозування для отриманого нелінійного рівняння регресії побудуємо з використанням побудованих для лінійного рівняння регресії інтервалів прогнозування та передбачення, а також зворотного нормалізуючого перетворення (2.3). У результаті отримаємо удосконалену нелінійну регресійну модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

2.2 Перевірка емпіричних даних на викиди

Для побудови удосконаленої регресійної моделі оцінювання розміру мобільних застосунків потрібно перевірити емпіричні дані на викиди. Для знаходження викидів використаємо квадрат відстані Махаланобіса (Mahalanobis squared distance) [20]:

$$d_i^2 = (Z_i - \overline{Z})^T S_N{}^{-1} (Z_i - \overline{Z}), \tag{2.7}$$

де $\overline{Z}$ – середній вектор вибірки;

S_N – матриця кореляції вибірки, що визначається за наступною формулою:

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

Якщо певна точка має найбільшу відстань Махаланобіса до всіх інших точок, то вважається, що вона має найбільшу значимість, оскільки найбільше впливає на коефіцієнти та кривизну рівняння регресії.

Маючи розраховані значення d^2, для визначення викидів можна використати критерій Пірсона χ^2 або критерій Фішера F. Для використання критерію Фішера F необхідно також розрахувати тестову статистику T_S окремо для кожного значення d_i^2. Тестова статистика для d_i^2 має апроксимований розподіл F з m та N-m ступенями свободи та розраховується за наступною формулою:

$$T_S = \frac{(N-m)Nd_i^2}{(N^2-1)m}. \tag{2.8}$$

У даному випадку будемо використовувати критерій Фішера F. Тестову статистику для квадрату відстані Махаланобіса T_S потрібно порівняти з квантилем розподілу F, який визначається як $F_{m,N-m,\alpha}$, де α – це рівень значущості, який у даному випадку дорівнюватиме 0,05. Якщо для певної точки значення T_S, розраховане за формулою (2.8), буде перевищувати квантиль розподілу F, то ця точка вважається викидом та видаляється з набору даних.

Отриманий набір даних нормалізується і знову перевіряється на викиди. Вказані дії повторюються ітераційно, доки всі викиди не будуть видалені, тобто всі значення T_S будуть менше або дорівнюватимуть квантилю розподілу F.

2.3 Побудова удосконаленої регресійної моделі для оцінювання розміру мобільних застосунків на Kotlin, та перевірка її якості

За критерієм χ^2 з використанням формули (2.1) визначено, що вихідні дані не підпорядковуються нормальному закону розподілу, оскільки $\chi^2_x = 40{,}8$, $\chi^2_y = 13{,}82$ при критичному значенні $\chi^2 = 7{,}81$, тому необхідно застосувати нормалізуючі перетворення на основі натурального логарифму [16]. Отримаємо нормалізовані значення z_x та z_y, які разом з емпіричними даними представлені в табл. 2.1.

Таблиця 2.1 – Емпіричні та нормалізовані значення випадкових величин

№	x	y	z_x	z_y
1	2	3	4	5
1	900	63745	6,8024	11,0626
2	396	27637	5,9814	10,2269
3	287	21457	5,6595	9,9738
4	185	10197	5,2204	9,2298
5	372	27721	5,9189	10,2299
6	54	1645	3,9890	7,4055
7	35	2957	3,5553	7,9919
8	767	61856	6,6425	11,0326
9	39	2139	3,6636	7,6681
10	556	40702	6,3208	10,6140
11	78	6181	4,3567	8,7292
12	247	11690	5,5094	9,3665
13	224	48641	5,4116	10,7922
14	192	21787	5,2575	9,9891
15	93	4105	4,5326	8,3200
16	358	17099	5,8805	9,7468
17	114	13081	4,7362	9,4789
18	1105	77156	7,0076	11,2536
19	171	18290	5,1417	9,8141
20	222	11315	5,4027	9,3339
21	456	27137	6,1225	10,2087
22	235	17529	5,4596	9,7716
23	33	706	3,4965	6,5596
24	975	69419	6,8824	11,1479
25	27	1449	3,2958	7,2786
26	461	27114	6,1334	10,2078
27	281	18319	5,6384	9,8157
28	103	16655	4,6347	9,7205
29	236	12071	5,4638	9,3986
30	1561	112761	7,3531	11,6330
31	2130	129250	7,6639	11,7695
32	608	49505	6,4102	10,8098
33	610	48093	6,4135	10,7809
34	280	21336	5,6348	9,9682
35	292	16780	5,6768	9,7279
36	72	4735	4,2767	8,4627
37	138	19221	4,9273	9,8638
38	4	370	1,3863	5,9135
39	139	11747	4,9345	9,3714
40	35	1273	3,5553	7,1491

Перевірка отриманих нормалізованих даних на наявність викидів проведена з використанням квадрату відстані Махаланобіса (2.7) та тестової статистики T_S (2.8). Отриману вибірку даних після усунення викидів представлено в табл. 2.2.

Таблиця 2.2 – Емпіричні та нормалізовані дані після усунення викидів

№	x	y	z_x	z_y
1	2	3	4	5
1	900	63745	6,8024	11,0626
2	396	27637	5,9814	10,2269
3	287	21457	5,6595	9,9738
4	185	10197	5,2204	9,2298
5	372	27721	5,9189	10,2299
6	35	2957	3,5553	7,9919
7	767	61856	6,6425	11,0326
8	39	2139	3,6636	7,6681
9	556	40702	6,3208	10,6140
10	78	6181	4,3567	8,7292
11	247	11690	5,5094	9,3665
12	192	21787	5,2575	9,9891
13	93	4105	4,5326	8,3200
14	358	17099	5,8805	9,7468
15	114	13081	4,7362	9,4789
16	1105	77156	7,0076	11,2536
17	171	18290	5,1417	9,8141
18	222	11315	5,4027	9,3339
19	456	27137	6,1225	10,2087
20	235	17529	5,4596	9,7716
21	975	69419	6,8824	11,1479
22	27	1449	3,2958	7,2786
23	461	27114	6,1334	10,2078
24	281	18319	5,6384	9,8157
25	236	12071	5,4638	9,3986
26	1561	112761	7,3531	11,6330
27	608	49505	6,4102	10,8098
28	610	48093	6,4135	10,7809
29	280	21336	5,6348	9,9682
30	292	16780	5,6768	9,7279
31	72	4735	4,2767	8,4627
32	4	370	1,386294	5,913503
33	139	11747	4,934474	9,371353

Розподіли для емпіричних та нормалізованих за допомогою натурального логарифму значень вибірки *X* представлено на рис. 2.1 та рис. 2.2 відповідно.

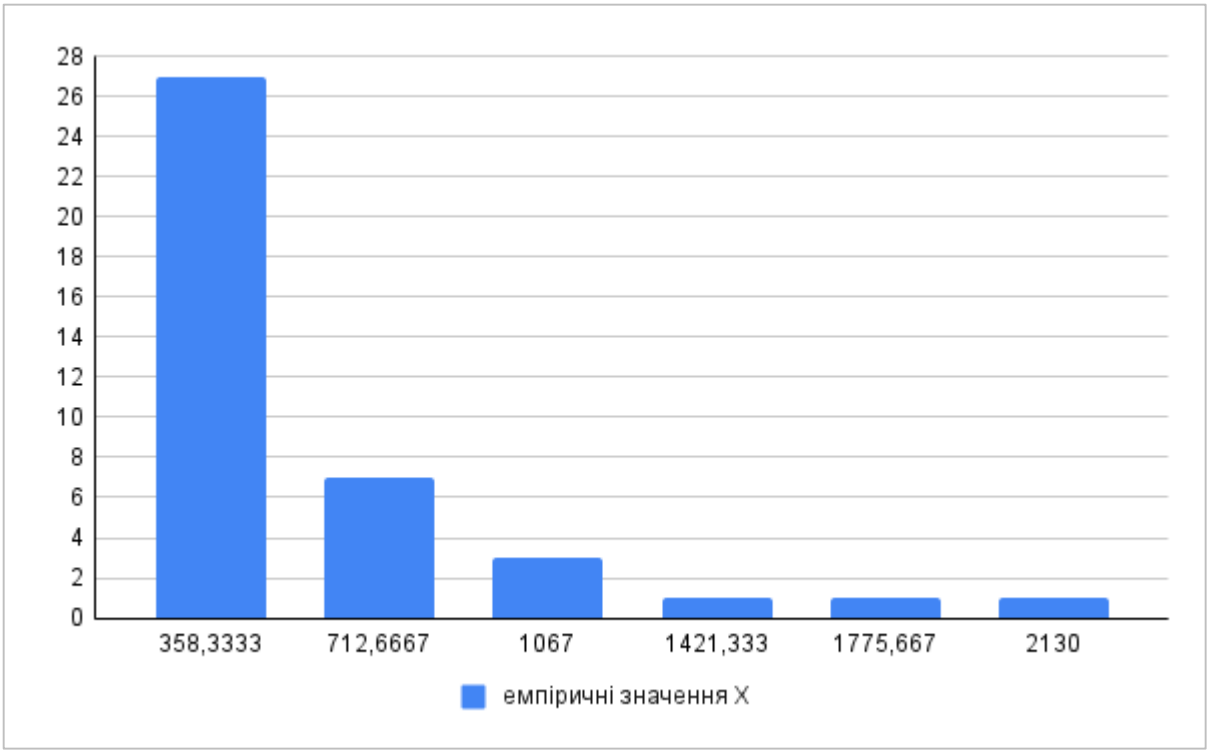

Рисунок 2.1 – Розподіл емпіричних даних для вибірки X

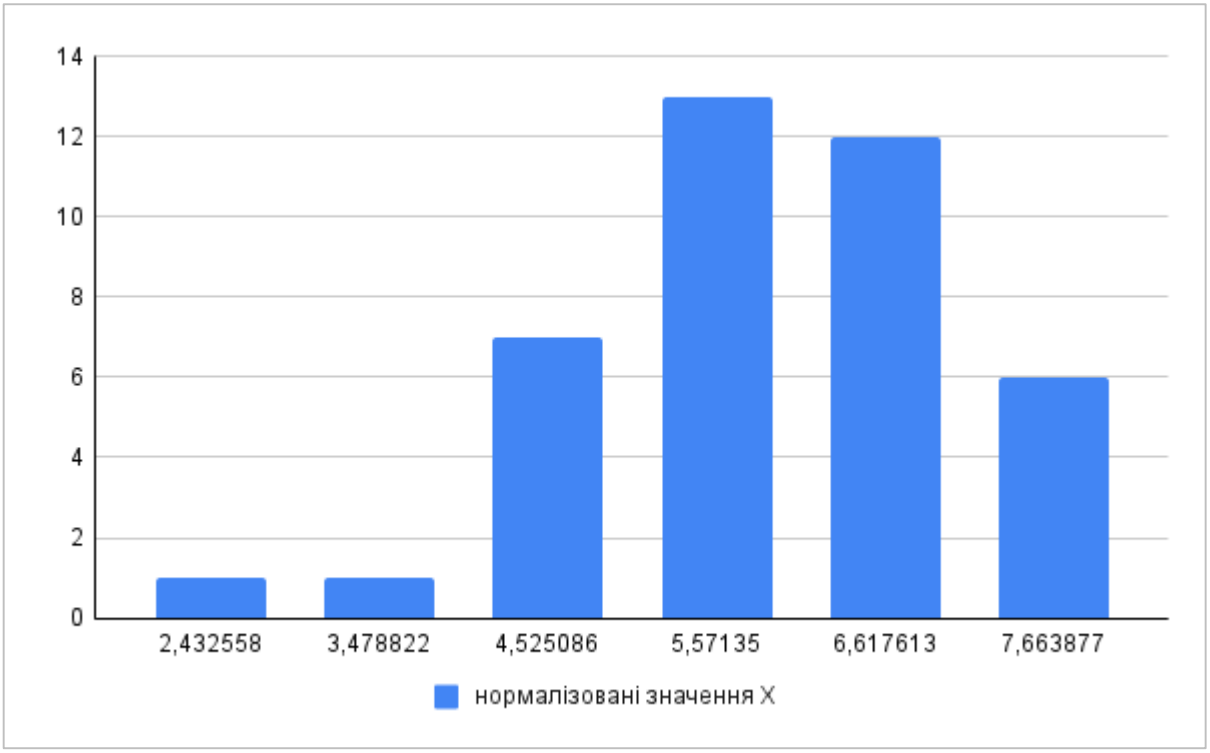

Рисунок 2.2 – Розподіл нормалізованих даних для вибірки X

Розподіли для емпіричних та нормалізованих за допомогою натурального логарифму даних вибірки Y представлено на рис. 2.3 та рис. 2.4 відповідно.

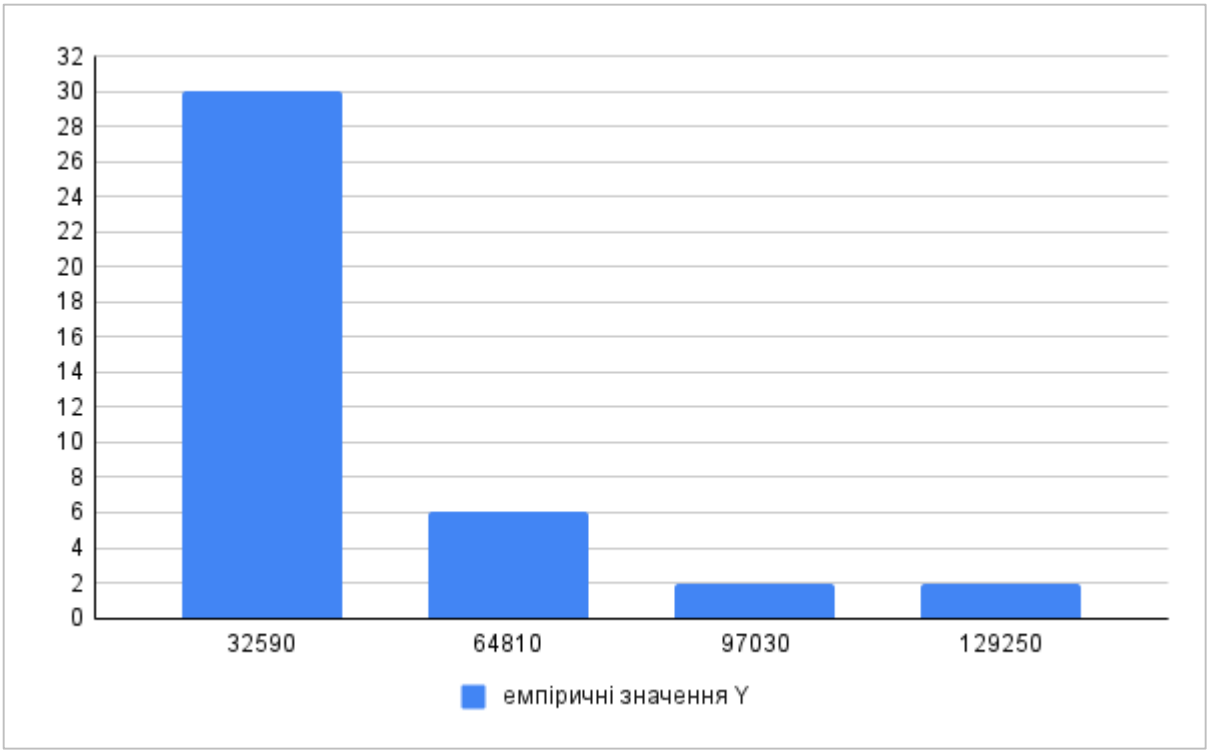

Рисунок 2.3 – Розподіл емпіричних даних для вибірки Y

Рисунок 2.4 – Розподіл нормалізованих даних для вибірки Y

Для побудови лінійного рівняння регресії розраховуємо значення b_0, b_1 за допомогою методу найменших квадратів відповідно з (2.5), отримуємо наступні значення:

$$b_0 = 0{,}9877,\ b_1 = 4{,}3058.$$

За допомогою отриманих коефіцієнтів відповідно з (2.4) будуємо лінійне рівняння регресії:

$$z_y = 4{,}3058 \cdot z_x + 0{,}9877 + \varepsilon.$$

Для отриманої лінійної регресійної моделі будуємо довірчий інтервал відповідно до (1.5) та інтервал прогнозування відповідно до (1.6), які разом з самим лінійним рівнянням регресії та нормалізованими випадковими величинами представлено на рис. 2.5

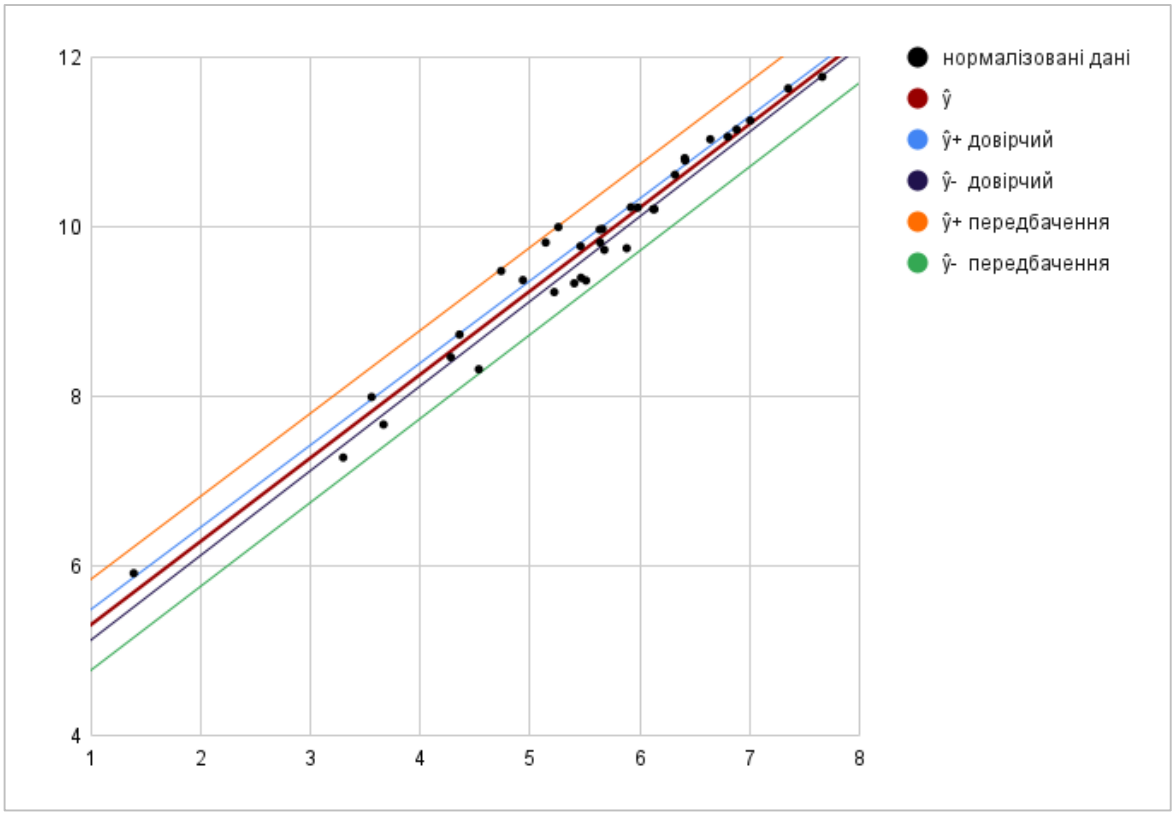

Рисунок 2.5 – Нормалізовані випадкові величини, лінійне рівняння регресії, довірчий інтервал та інтервал передбачення

Оскільки на рис. 2.5 видно, що всі нормалізовані випадкові величини знаходяться в межах інтервалу прогнозування, а також $\chi^2_x = 1{,}78$, $\chi^2_y = 2{,}97$ при критичному значенні $\chi^2 = 7{,}81$, то можна зробити висновок, що дані підпорядковуються нормальному закону розподілу та не містять викидів.

Для побудови нелінійної регресійної моделі переходимо до вихідних даних та на основі лінійного рівняння регресії з використанням зворотного нормалізуючого перетворення (2.3) будуємо нелінійну регресію згідно з (2.6):

$$y = e^{4{,}3058+\varepsilon} \cdot x^{0{,}9877}.$$

За формулами зворотного перетворення (2.3) розраховуємо границі довірчого інтервалу та інтервалу прогнозування для побудованої нелінійної регресійної моделі. На рис. 2.6 представлено побудовану нелінійну регресійну модель, довірчий інтервал, інтервал прогнозування, а також емпіричні дані.

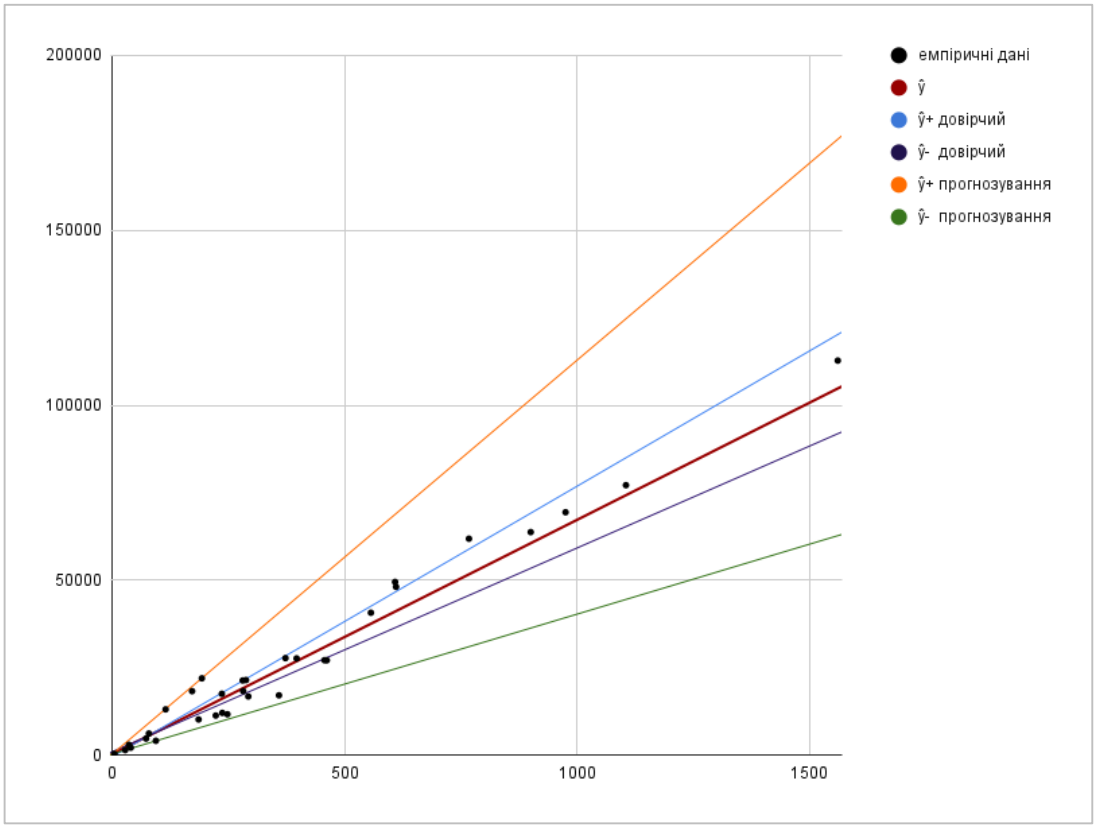

Рисунок 2.6 – Емпіричні дані, нелінійне рівняння регресії, довірчий інтервал та інтервал передбачення

Для порівняння з удосконаленою нелінійною регресійною моделлю будуємо лінійне рівняння регресії у припущенні про нормальність розподілу емпіричних даних. Розраховуємо коефіцієнти за допомогою методу найменших квадратів (2.5) і отримуємо наступне лінійне рівняння регресії:

$$z_y = 72{,}36 \cdot z_x - 498{,}10 + \varepsilon. \tag{2.9}$$

Побудуємо довірчий інтервал та інтервал прогнозування для лінійного рівняння регресії згідно з (1.5) та (1.6) відповідно. Побудоване у припущенні про гаусівський розподіл емпіричних даних лінійне рівняння регресії, самі емпіричні дані, довірчий інтервал та інтервал прогнозування представлені на рис. 2.7.

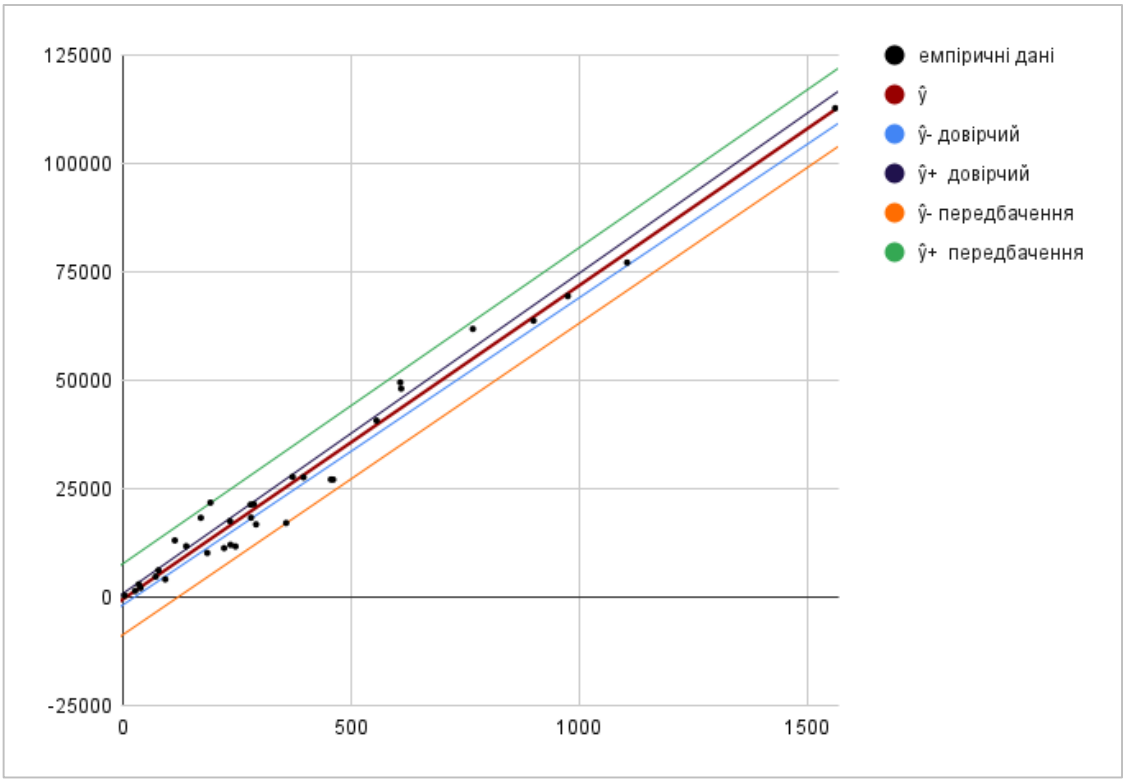

Рисунок 2.7 – Емпіричні дані, лінійне рівняння регресії у припущенні про нормальність розподілу, довірчий інтервал та інтервал передбачення для ненормалізованих даних

З рис. 2.7 видно, що нижня границя інтервалу прогнозування лінійного рівняння регресії містить від`ємні значення, на відміну від удосконаленої нелінійної регресійної моделі, зображеної на рис. 2.6.

Випадкова похибка ε для лінійного рівняння регресії має наступні показники: математичне сподівання $m_\varepsilon = 0$, дисперсія $S_\varepsilon = 0{,}0319$, $\chi^2_\varepsilon = 28{,}13$ при критичному значенні $\chi^2_{кр} = 5{,}99$. Оскільки $\chi^2_\varepsilon > \chi^2_{кр}$, то значення випадкової похибки ε не підпорядковуються нормальному закону розподілу, тому у даному випадку використовувати лінійне рівняння регресії у припущенні про нормальність розподілу емпіричних даних не бажано, оскільки це може призвести до спотворених результатів оцінювання розміру мобільних застосунків.

Також для перевірки якості удосконаленої нелінійної регресійної моделі розрахуємо такі показники як коефіцієнт детермінації R^2, середню величину відносної похибки MMRE та рівень прогнозування PRED(l) при $l = 0{,}25$. Порівняємо перелічені показники якості для побудованої нелінійної регресійної моделі з використанням нормалізуючого перетворення на основі натурального логарифму та лінійного рівняння регресії, побудованого з припущенням про нормальність розподілу емпіричних даних. Показники якості для лінійної та нелінійної регресійних моделей наведено в табл. 2.3.

Таблиця 2.3 – Оцінка якості лінійної та нелінійної регресійних моделей

Показник	R^2	MMRE	PRED(0,25)
Лінійне рівняння регресії	0,9419	0,2228	0,6667
Нелінійне рівняння регресії	0,9743	0,1964	0,6765

Із наведених даних бачимо, що значення показників якості регресійних моделей покращені для нелінійного рівняння регресії. Значення коефіцієнту детермінації R^2 вище на 0,0324, значення середньої величини відносної похибки MMRE менше на 0,0264, рівень прогнозування PRED(0,25) перевищує на 0,0098.

У табл. 2.4 представлено границі довірчого інтервалу та інтервалу передбачення, а також прогнозовані значення за лінійним рівняння регресії (2.9).

Таблиця 2.4 – Межі довірчого інтервалу та інтервалу передбачення, прогнозовані значення за лінійним рівняння регресії для емпіричних даних

№	Результат прогнозу	Нижня границя довірчого інтервалу	Верхня границя довірчого інтервалу	Нижня границя інтервалу прогнозування	Верхня границя інтервалу прогнозування
1	2	3	4	5	6
1	64624,57	62051,35	67197,80	56080,25	73168,90
2	28155,88	26734,81	29576,95	19885,24	36426,52
3	20268,80	18806,27	21731,32	11990,94	28546,66
4	12888,23	11272,70	14503,76	4581,97	21194,49
5	26419,27	25000,92	27837,63	18149,10	34689,45
6	2034,45	51,23	4017,67	-6351,08	10419,98
7	55000,89	52859,91	57141,87	46576,65	63425,13
8	2323,89	352,05	4295,72	-6058,96	10706,73
9	39733,24	38132,59	41333,90	31429,86	48036,62
10	5145,87	3281,18	7010,56	-3212,43	13504,16
11	17374,46	15863,73	18885,18	9087,95	25660,97
12	13394,74	11792,71	14996,77	5091,09	21698,38
13	6231,25	4405,74	8056,75	-2118,39	14580,89
14	25406,25	23986,36	26826,15	17135,82	33676,69
15	7750,77	5978,02	9523,53	-587,49	16089,04
16	79458,07	76153,78	82762,37	70665,89	88250,25
17	11875,21	10231,54	13518,88	3563,43	20186,99
18	15565,50	14016,64	17114,35	7271,95	23859,05
19	32497,39	31040,40	33954,38	24220,50	40774,28
20	16506,16	14977,86	18034,45	8216,42	24795,89
21	70051,46	67217,59	72885,34	61425,06	78677,87
22	1455,58	-550,61	3461,78	-6935,41	9846,58
23	32859,18	31397,38	34320,99	24581,45	41136,92
24	19834,65	18365,95	21303,35	11555,69	28113,60
25	16578,51	15051,74	18105,29	8289,06	24867,97
26	112453,56	107403,28	117503,84	102867,67	122039,45
27	43495,89	41786,39	45205,38	35170,84	51820,93
28	43640,60	41926,53	45354,67	35314,62	51966,59
29	19762,29	18292,53	21232,05	11483,15	28041,43
30	20630,59	19172,92	22088,27	12353,59	28907,60
31	4711,72	2831,02	6592,41	-3650,16	13073,60

Продовження табл. 2.4

1	2	3	4	5	6
32	-208,66	-2282,37	1865,04	-8616,06	8198,73
33	9559,74	7846,29	11273,18	1233,88	17885,60

У табл. 2.5 представлено межі довірчого інтервалу, інтервалу прогнозування та прогнозовані значення за нелінійним рівняння регресії, побудованим з використанням нормалізуючого перетворення на основі натурального логарифму.

Таблиця 2.5 – Межі довірчого інтервалу та інтервалу передбачення, прогнозовані значення за нелінійним рівняння регресії для нормалізованих даних

№	Результат прогнозу	Нижня границя довірчого інтервалу	Верхня границя довірчого інтервалу	Нижня границя інтервалу прогнозування	Верхня границя інтервалу прогнозування
1	2	3	4	5	6
1	61345,44	53663,20	70127,44	36414,55	103345,03
2	27266,62	24746,73	30043,10	16318,86	45558,83
3	19840,01	18140,61	21698,61	11890,14	33105,25
4	12858,29	11764,77	14053,46	7706,88	21452,98
5	25633,85	23307,85	28191,97	15346,97	42815,90
6	2483,11	2113,38	2917,53	1462,66	4215,49
7	52383,13	46216,69	59372,32	31160,76	88059,21
8	2763,20	2367,21	3225,44	1630,84	4681,82
9	38123,58	34160,11	42546,91	22758,06	63863,39
10	5479,37	4875,78	6157,68	3265,97	9192,83
11	17106,48	15665,06	18680,54	10254,64	28536,52
12	13338,71	12209,08	14572,86	7995,37	22253,02
13	6518,95	5847,49	7267,50	3892,40	10917,84
14	24680,81	22464,68	27115,54	14779,22	41216,12
15	7970,93	7207,56	8815,15	4767,15	13327,83
16	75128,23	64962,01	86885,43	44458,15	126956,50
17	11896,77	10873,20	13016,70	7129,22	19852,53
18	15395,30	14101,85	16807,39	9229,28	25680,79
19	31343,35	28310,70	34700,87	18741,38	52419,07
20	16285,40	14916,30	17780,16	9762,79	27165,82
21	66392,00	57819,98	76234,86	39364,53	111976,38
22	1921,68	1609,36	2294,61	1126,15	3279,17
23	31682,77	28605,71	35090,82	18942,81	52990,96
24	19430,30	17771,21	21244,28	11645,20	32419,94

Продовження табл. 2.5

1	2	3	4	5	6
25	16353,84	14978,81	17855,10	9803,80	27280,05
26	105680,23	89504,42	124779,45	62156,12	179681,63
27	41643,16	37163,81	46662,40	24837,43	69820,13
28	41778,45	37278,88	46821,13	24917,29	70049,32
29	19362,00	17709,58	21168,61	11604,36	32305,72
30	20181,36	18447,94	22077,65	12094,16	33676,35
31	5062,88	4487,55	5711,97	3015,01	8501,70
32	291,48	214,66	395,79	161,63	525,64
33	9695,21	8821,49	10655,47	5805,25	16191,75

З наведених таблиць бачимо, що довжини довірчих інтервалів та інтервалів передбачення для удосконаленого нелінійного рівняння регресії переважно є меншими ніж для лінійної регресійної моделі, побудованої у припущенні про нормальність розподілу емпіричних даних.

Також оскільки значення показників якості для нелінійної регресійної моделі кращі ніж при використанні лінійної моделі, то нелінійна регресійна модель для прогнозування розміру мобільних застосунків, що створюються на Kotlin, була удосконалена. Удосконалена регресійна модель, побудована з використанням нормалізуючого перетворення за допомогою натурального логарифму, може використовуватись для передбачення розміру мобільних застосунків.

2.4 Постановка задачі на розробку програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin

У результаті аналізу існуючих методів та моделей для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, можна сформулювати наступну постановку задачі на розробку програмного забезпечення.

У даній роботі необхідно розробити програмне забезпечення, яке буде виконувати наступні функції:

- внесення даних про проекти мобільних застосунків (кількість рядків коду, кількість класів);
- виконання нормалізації внесених даних із застосуванням перетворення на основі натурального логарифму;
- побудова довірчого інтервалу на основі нормалізованих даних;
- побудова інтервалу передбачення на основі нормалізованих даних;
- побудова нелінійної регресійної моделі на основі лінійного рівняння регресії;
- оцінювання розміру мобільних застосунків.

Детальні вимоги до програмного забезпечення наведено у додатку А.

3 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОБІЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN

Під час розробки проекту програмного забезпечення, який складається з ескізного, технічного та робочого проектів, було використано об’єктно-орієнтовану методологію та уніфіковану мову моделювання UML.

3.1 Ескізній проект програмного забезпечення для оцінювання розміру мобільних застосунків

На етапі ескізного проектування програмного забезпечення «Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin» за допомогою мови моделювання UML була побудована діаграма варіантів використання та розроблені специфікації варіантів використання. Також для основних прецедентів були розроблені сценарії, які представлені у вигляді діаграм діяльності. Згідно з побудованими сценаріями було розроблено проект користувацького інтерфейсу.

3.1.1 Побудова моделі варіантів використання

Діаграма варіантів використання розробляється з метою визначення основних вимог до програмного продукту. Ця діаграма описує функціональність системи у вигляді прецедентів, а також діючих осіб, що взаємодіють з системою, у вигляді акторів. Розроблена діаграма варіантів використання дозволяє отримати уявлення про функціональну поведінку системи та її взаємодію з користувачем.

У даному випадку в результаті аналізу вимог до програмного забезпечення було виділену діючу особу або актора - користувача. Виявлених акторів програмного забезпечення представлено в табл. 3.1.

Таблиця 3.1 – Актори програмного забезпечення

Актор	Короткий опис
Користувач	Особа, яка взаємодіє з програмним забезпеченням та має наступні можливості: введення вхідних даних, виконання нормалізації введених даних, побудова лінійної та нелінійної регресії, обчислення довірчого інтервалу та інтервалу передбачення, обчислення показників якості регресійних моделей, оцінка розміру

У табл. 3.2 наведено короткий опис варіантів використання, які відображають функції розроблюваного програмного забезпечення.

Таблиця 3.2 – Опис варіантів використання програмного забезпечення

Найменування варіанту використання	Короткий опис
Введення вхідних даних	Варіант використання надає можливість користувачеві ввести вхідні з обраного файлу
Перетворення вхідних даних	Варіант використання надає можливість нормалізувати вхідні дані за допомогою натурального логарифму
Побудова нелінійної регресії	Варіант використання надає можливість побудови нелінійної регресії
Побудова лінійної регресії	Варіант використання надає можливість побудови лінійної регресії
Побудова довірчого інтервалу	Варіант використання надає можливість користувачеві отримати розрахунок довірчого інтервалу регресійної моделі
Побудова інтервалу передбачення	Варіант використання надає можливість користувачеві отримати розрахунок інтервалу передбачення регресійної моделі
Виконати оцінювання регресії	Варіант використання надає можливість користувачеві розрахувати наступні показники якості регресійної моделі: R^2, MMRE, PRED(0,25)
Передбачення розміру	Варіант використання надає можливість оцінювання розміру за допомогою нелінійної регресії

Розроблену діаграму варіантів використання для програмного забезпечення "Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin" представлено на рис. 3.1

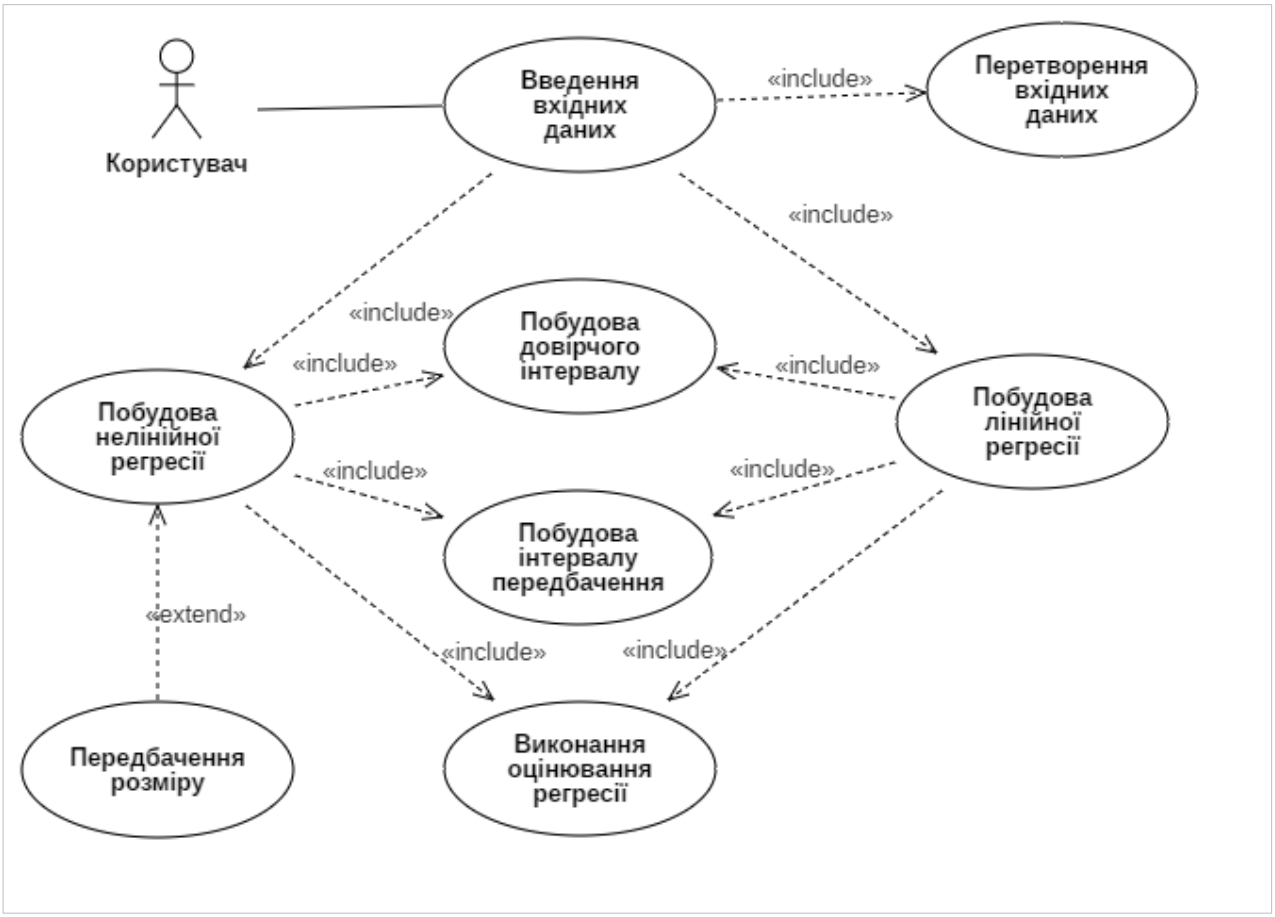

Рисунок 3.1 – Діаграма варіантів використання програмного забезпечення

Для розроблених варіантів використання, наведених на рис. 3.1, необхідно привести специфікації варіантів використання для формалізації функціональних вимог до програмного забезпечення, що розробляється.

3.1.2 Специфікації варіантів використання

В табл. 3.3 наведено опис специфікації варіанту використання «Введення вхідних даних».

Таблиця 3.3 – Опис специфікації варіанту використання «Введення вхідних даних»

Розділ	Опис
1	2
Найменування	Введення вхідних даних
Короткий опис	Варіант використання надає можливість ввести вхідні дані у програму
Передумови	Користувач має запустити програму та натиснути на кнопку для введення даних
Основний потік	Користувач обирає файл з вхідними даними
Постумови	Система надає можливість виконання варіантів використання «Перетворення вхідних даних» та «Побудова лінійної регресії»

В табл. 3.4 наведено опис специфікації варіанту використання «Перетворення вхідних даних».

Таблиця 3.4 – Опис специфікації варіанту використання «Перетворення вхідних даних»

Розділ	Опис
1	2
Найменування	Перетворення вхідних даних
Короткий опис	Варіант використання надає можливість здійснити нормалізацію вхідних даних
Передумови	Варіант використання доступний після виконання «Введення вхідних даних»
Основний потік	Система здійснює нормалізацію вхідних даних за допомогою натурального логарифму
Постумови	Система надає можливість виконання варіанту використання «Побудова нелінійної регресії»

В табл. 3.5 наведено опис специфікації варіанту використання «Побудова нелінійної регресії».

Таблиця 3.5 – Опис специфікації варіанту використання «Побудова нелінійної регресії»

Розділ	Опис
1	2
Найменування	Побудова нелінійної регресії
Короткий опис	Варіант використання надає можливість побудувати нелінійну регресію
Передумови	Варіант використання доступний після виконання «Перетворення вхідних даних»
Основний потік	Система будує лінійну регресію
Постумови	Виведення побудованої лінійної регресії на екран, а також система надає можливість виконання варіанту використання «Передбачення розміру»

В табл. 3.6 наведено опис специфікації варіанту використання «Побудова лінійної регресії».

Таблиця 3.6 – Опис специфікації варіанту використання «Побудова лінійної регресії»

Розділ	Опис
1	2
Найменування	Побудова лінійної регресії
Короткий опис	Варіант використання надає можливість побудувати лінійну регресію
Передумови	Варіант використання доступний після виконання «Введення вхідних даних»
Основний потік	Система будує лінійну регресію
Постумови	Виведення побудованої лінійної регресії на екран, а також система надає можливість виконання варіанту використання «Побудова нелінійної регресії»

В табл. 3.7 наведено опис специфікації варіанту використання «Побудова довірчого інтервалу».

Таблиця 3.7 – Опис специфікації варіанту використання «Побудова довірчого інтервалу»

Розділ	Опис
1	2
Найменування	Побудова довірчого інтервалу
Короткий опис	Варіант використання надає можливість побудувати довірчий інтервал
Передумови	Варіант використання доступний після виконання «Побудова лінійної регресії» або «Побудова нелінійної регресії»
Основний потік	Система будує довірчий інтервал
Постумови	Відсутні

В табл. 3.8 наведено опис специфікації варіанту використання «Побудова інтервалу передбачення».

Таблиця 3.8 – Опис специфікації варіанту використання «Побудова інтервалу передбачення»

Розділ	Опис
1	2
Найменування	Побудова інтервалу передбачення
Короткий опис	Варіант використання надає можливість побудувати інтервал передбачення
Передумови	Варіант використання доступний після виконання «Побудова лінійної регресії» або «Побудова нелінійної регресії»
Основний потік	Система будує інтервал передбачення
Постумови	Відсутні

В табл. 3.9 наведено опис специфікації варіанту використання «Виконати оцінювання регресії».

Таблиця 3.9 – Опис специфікації варіанту використання «Виконання оцінювання регресії»

Розділ	Опис
1	2
Найменування	Виконання оцінювання регресії
Короткий опис	Варіант використання надає можливість розрахунку показників якості регресійної моделі
Передумови	Варіант використання доступний після виконання «Побудова лінійної регресії» або «Побудова нелінійної регресії»
Основний потік	Система розраховує наступні показники якості регресії: R^2, MMRE, PRED(0,25)
Постумови	Виведення розрахованих показників на екран

В табл. 3.10 наведено опис специфікації варіанту використання «Передбачення розміру».

Таблиця 3.10 – Опис специфікації варіанту використання «Передбачення розміру»

Розділ	Опис
1	2
Найменування	Передбачення розміру
Короткий опис	Варіант використання надає можливість розрахунку показників якості регресійної моделі
Передумови	Варіант використання доступний після виконання «Побудова нелінійної регресії»
Основний потік	Система розраховує оцінку розміру на основі нелінійної регресійної моделі
Постумови	Виведення розрахованої оцінки на екран

Таким чином, для варіантів використання були розроблені специфікації для формалізації функціональних вимог до програмного забезпечення.

3.1.3 Побудова діаграм діяльності для основних варіантів використання

Для більш точного відображення послідовності дій, які здійснюються у процесі виконання варіанту використання, створюється діаграма діяльності. Діаграми діяльності були розроблені для основних варіантів використання, наведених на рис. 3.1.

Для варіанту використання «Побудова лінійної регресії» була розроблена діаграма діяльності, що представлена на рис. 3.2.

Рисунок 3.2 – Діаграма діяльності «Побудова лінійної регресії»

Для варіанту використання «Передбачення розміру» була розроблена діаграма діяльності, що представлена на рис. 3.3.

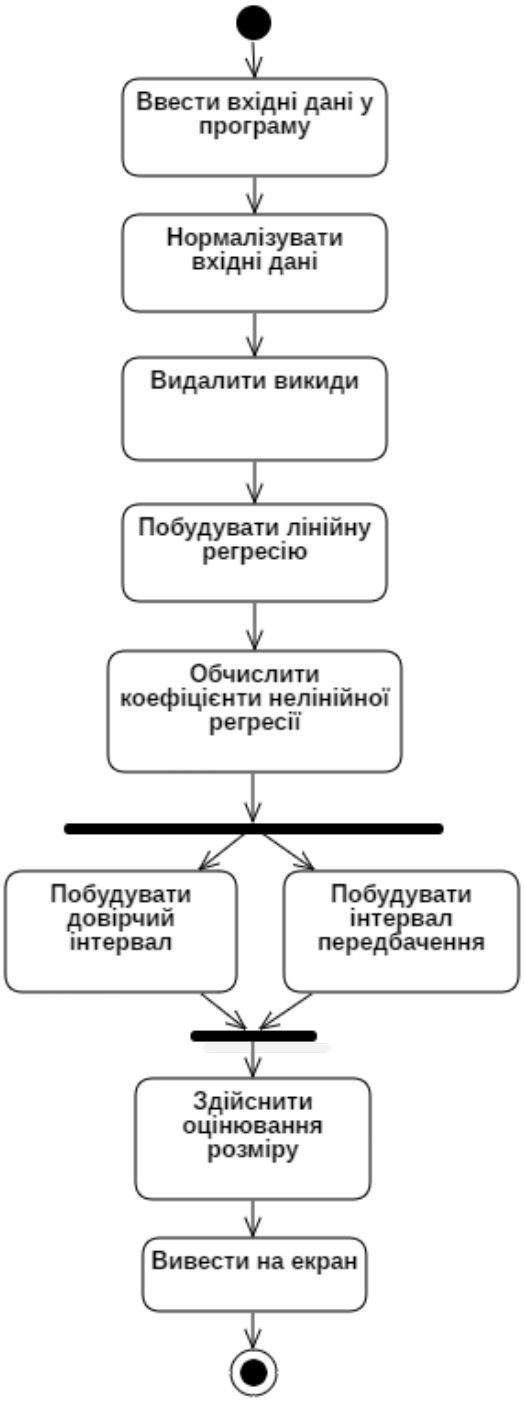

Рисунок 3.3 – Діаграма діяльності «Передбачення розміру»

Таким чином, для формалізації сценаріїв основних варіантів використання були розроблені діаграми діяльності.

3.1.4 Інформаційна модель

На основі аналізу вимог до розроблюваного програмного забезпечення для оцінювання розміру мобільних застосунків необхідно розробити інформаційну модель даних, яка відображає основні сутності, їх атрибути та взаємозв'язки.

Інформаційну модель програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, наведено на рис. 3.4.

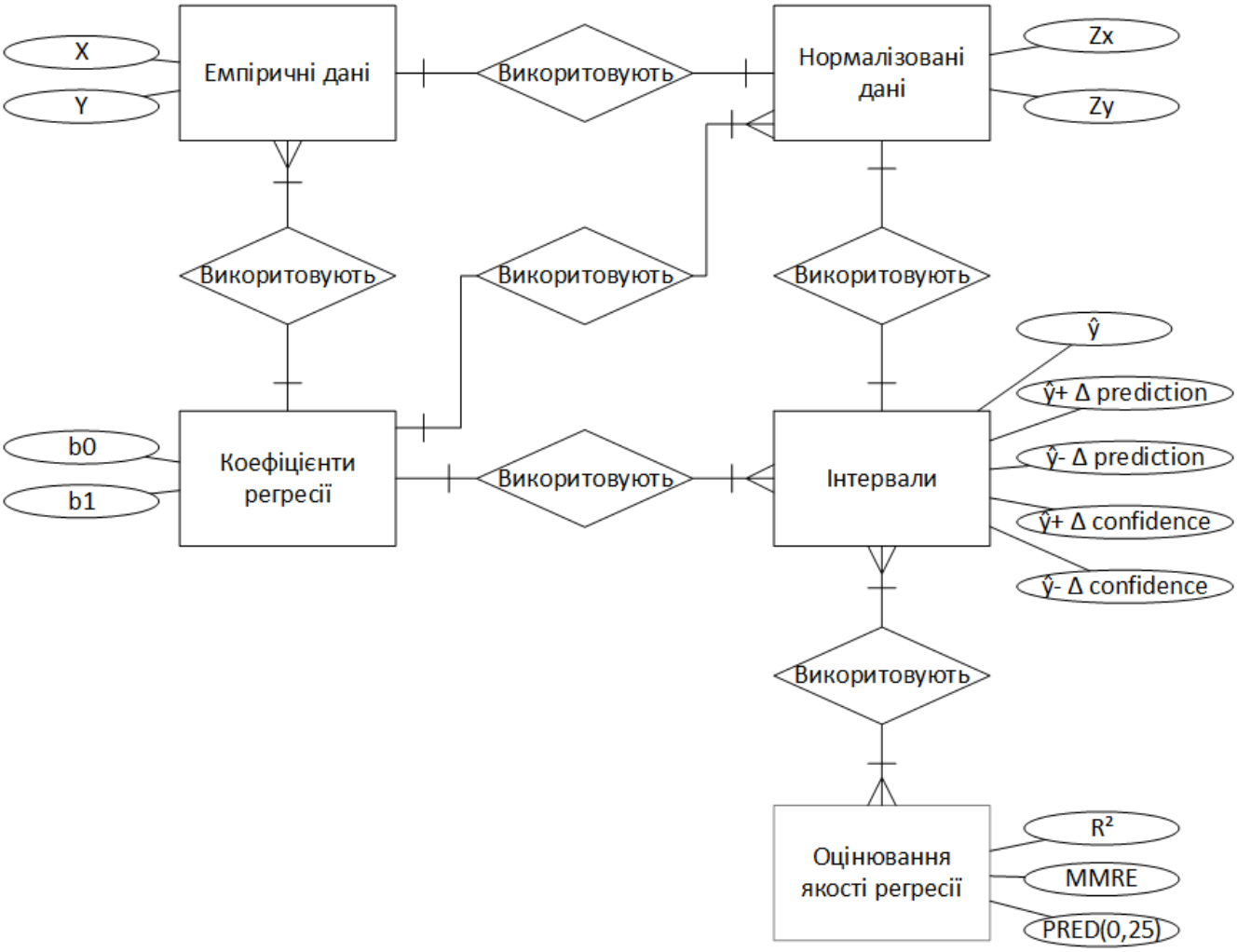

Рисунок 3.4 – Інформаційна модель програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin

Таким чином була розроблена інформаційна модель програмного забезпечення у вигляді ER-діаграми.

3.1.5 Проектування інтерфейсу користувача

Для взаємодії користувача з розроблюваним програмним забезпеченням необхідно розробити інтерфейс користувача. На основі визначених вимог розробимо зовнішній користувацький інтерфейс головної форми програмного забезпечення, що представлена на рис. 3.5.

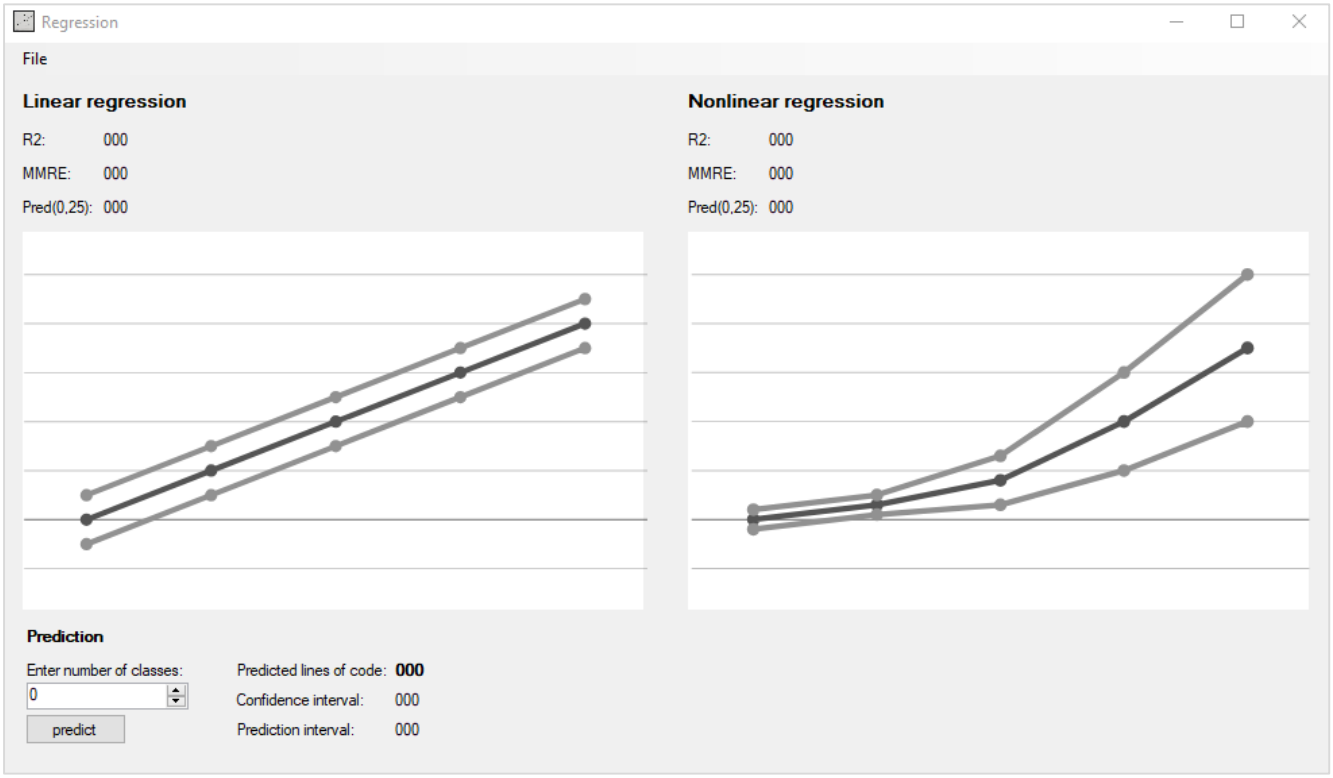

Рисунок 3.5 – Інтерфейс програмного забезпечення

Таким чином для програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, був розроблений інтерфейс користувача.

3.2 Технічний проект програмного забезпечення для оцінювання розміру мобільних застосунків

На основі ескізного проекту програмного забезпечення був розроблений технічний проект, який містить статичну модель у вигляді діаграми класів та динамічну модель у вигляді діаграми послідовності.

3.2.1 Побудова статичної моделі програмного забезпечення

На основі побудованої в ескізному проекті інформаційної моделі програмного забезпечення була побудована статична модель, що представлена діаграмою класів та наведена на рис. 3.6.

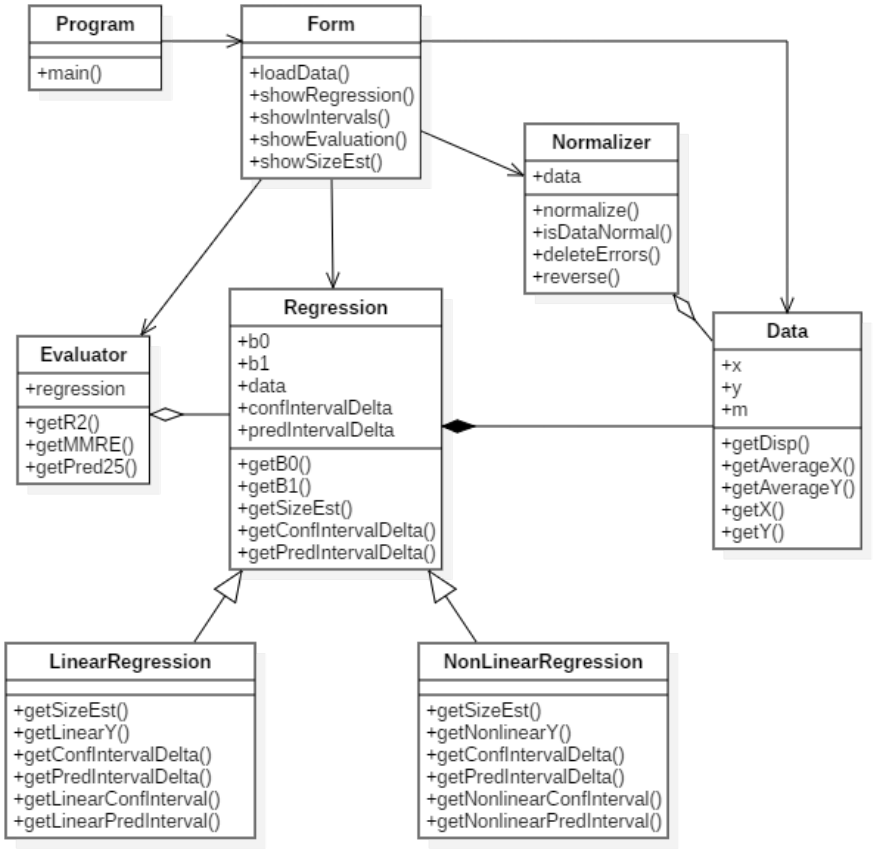

Рисунок 3.6 – Статична модель програмного забезпечення

Таким чином статична модель була розроблена у вигляді діаграми класів.

3.2.2 Специфікації класів

Для класів, наведених у статичній моделі програмного забезпечення, необхідно розробити специфікації, які відображають стан об’єкта та його поведінку.

Клас: «Program»;

Призначення: клас відповідальний за запуск програми.

Методи:

- main() – метод відповідальний за запуск програми.

Клас: «Form».

Призначення: відповідальний за відображення розрахунків на екран.

Методи:

- loadData() – завантажує файл з вхідними емпіричними даними;
- showRegression() – відображає побудовану регресійну модель;
- showIntervals() – відображає довірчий інтервал та інтервал передбачення регресійної моделі;
- showEvaluation() – відображає оцінки показників якості регресійної моделі;
- showSizeEst() – відображає розрахунок оцінки розміру мобільного застосунку.

Клас: «Data».

Призначення: містить вибірку випадкових величин.

Атрибути:

- x – кількість класів;
- y – кількість рядків коду;
- m – кількість підінтервалів, на які розбивається вибірка.

Методи:

- getDisp() – розраховує значення дисперсії вибірки;

- getAverageX() – розраховує середнє вибіркове значення величини x;
- getAverageY() – розраховує середнє вибіркове значення величини y;
- getX() – повертає значення x за вказаним індексом;
- getY() – повертає значення y за вказаним індексом.

Клас: «Regression».

Призначення: клас відповідальний за розрахунок коефіцієнтів регресії та побудову інтервалів.

Атрибути:

- b0 – перший коефіцієнт регресійної моделі;
- b1 – другий коефіцієнт регресійної моделі;
- data – вибірка випадкових величин;
- confIntervalDelta – границя довірчого інтервалу;
- predIntervalDelta – границя інтервалу передбачення;

Методи:

- getB0() – метод для отримання коефіцієнту регресії b0;
- getB1() – метод для отримання коефіцієнту регресії b1;
- getSizeEst() – повертає оцінку розміру мобільного застосунку;
- getConfIntervalDelta() – повертає границю довірчого інтервалу;
- getPredIntervalDelta() – повертає границю інтервалу передбачення.

Клас: «Normalizer».

Призначення: клас відповідальний за нормалізацію вхідних даних та зворотне перетворення.

Атрибути:

- data – екземпляр класу Data, який містить вибірку випадкових величин.

Методи:

- normalize() – нормалізовує вхідні дані;
- isDataNormal() – перевіряє нормальність розподілу даних;
- deleteErrors() – видалення викидів;

- reverse() – застосовує зворотне нормалізуюче перетворення.

Клас: «Evaluator».

Призначення: клас відповідальний за розрахунок показників якості регресійної моделі.

Атрибути:

- regression – екземпляр класу Regression.

Методи:

- getR2() – повертає розраховане значення R2;
- getMMRE() – повертає розраховане значення MMRE;
- getPred25() – повертає розраховане значення Pred(0,25).

Клас: «LinearRegression».

Призначення: клас відповідальний за розрахунок даних лінійної регресійної моделі.

Методи:

- getSizeEst() – повертає розрахункове значення y;
- getLinearY() – повертає розрахункове значення y за лінійним рівнянням регресії;
- getConfIntervalDelta() – повертає границю довірчого інтервалу;
- getPredIntervalDelta() – повертає границю інтервалу передбачення;
- getLinearConfInterval() – повертає границю довірчого інтервалу для лінійної регресії;
- getLinearPredInterval() – повертає границю інтервалу передбачення для лінійної регресії.

Клас: «NonlinearRegression».

Призначення: клас відповідальний за розрахунок даних лінійної регресійної моделі.

Методи:

- getSizeEst() – повертає оцінку розміру мобільного застосунку;
- getNonlinearY() – повертає розрахункове значення у за нелінійним рівнянням регресії;
- getConfIntervalDelta() – повертає границю довірчого інтервалу;
- getPredIntervalDelta() – повертає границю інтервалу передбачення;
- getNonlinearConfInterval() – повертає границю довірчого інтервалу для нелінійної регресії;
- getNonlinearPredInterval() – повертає границю інтервалу передбачення для нелінійної регресії.

3.2.3 Динамічна модель програмного забезпечення

На основі статичної моделі у вигляді діаграми класів та сценаріїв варіантів використання у вигляді діаграм діяльності необхідно створити динамічну модель програмного забезпечення у вигляді діаграми послідовності. Динамічна модель надає змогу отримати уявлення про взаємодію користувача з розроблюваною системою, а також про взаємодію між внутрішніми елементами системи.

Розроблена динамічна модель програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, представлена діаграмою послідовності та наведена на рис. 3.7.

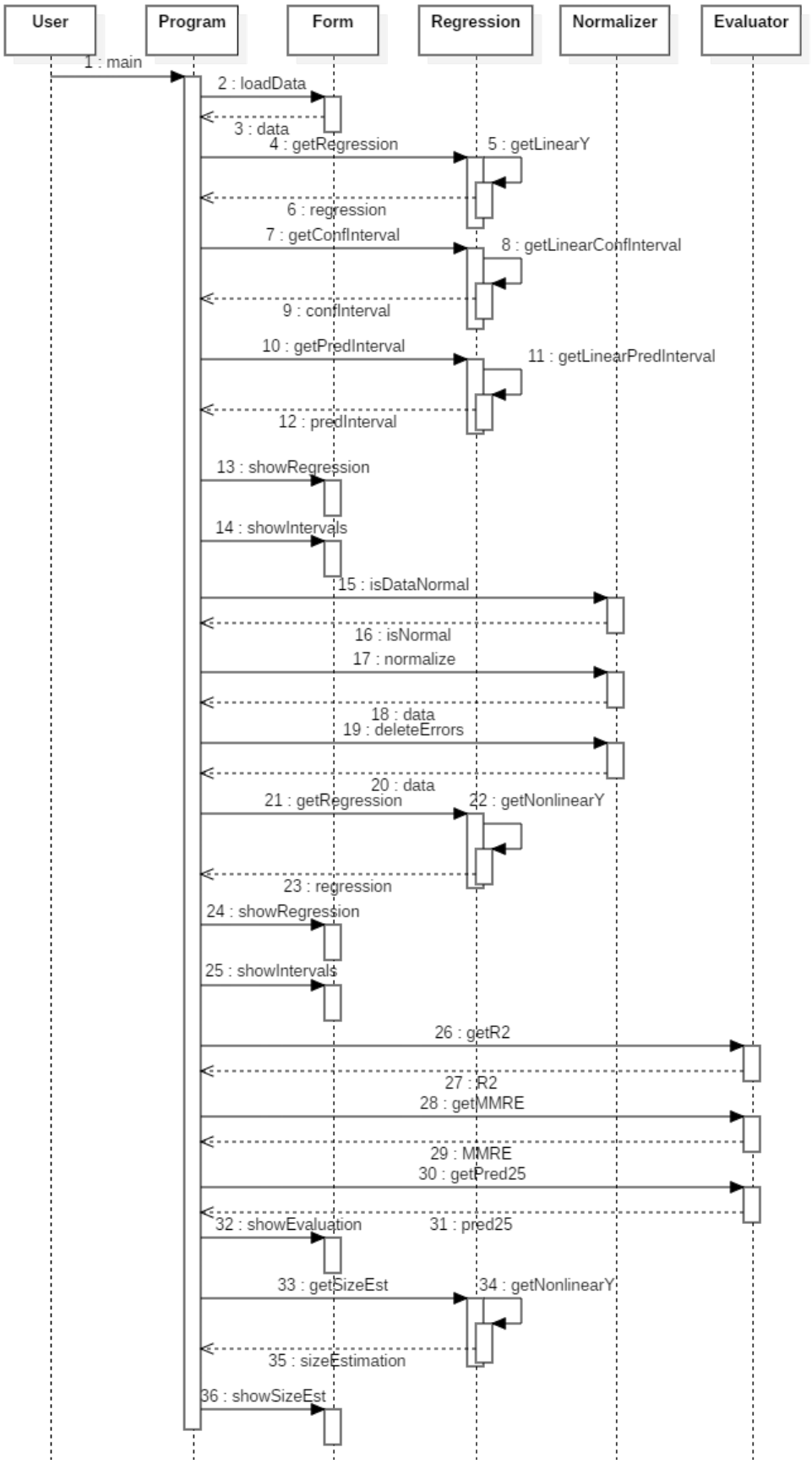

Рисунок 3.7 – Динамічна модель програмного забезпечення

Таким чином, динамічна модель була представлена діаграмою послідовності.

3.3 Робочий проект програмного забезпечення для оцінювання розміру мобільних застосунків

На етапі робочого проектування було обрано мову розробки програмного забезпечення для оцінювання розміру мобільних застосунків, а також середовище розробки. Розроблено діаграму компонентів програмного забезпечення, здійснено кодування, тестування та випробування розробленого програмного забезпечення.

3.3.1 Обґрунтування вибору мови програмування та середовища розробки програмного забезпечення

Для розробки програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, була обрана мова програмування C#. Мова C# була розроблена корпорацією Microsoft, а саме Андерсом Гейлсбергом, Пітером Гольде, Скотом Вілтамутом. C# має синтаксис, близький до своїх попередників C++ і Java, але на відміну від них виключає деякі недоліки, пов'язані з безпечністю. Наприклад, мова C# не підтримує множинне спадкування класів, яке було реалізоване у мові C++ та зарекомендувало себе як проблематичне.

Серед основних переваг мови C# можна виділити наступні:

- об'єктно-орієнтована;
- підтримує строгу статичну типізацію;
- більша надійність і простота синтаксису у порівнянні з мовами-попередниками такими як C та C++;
- можливість працювати з віртуальною машиною;
- можливість працювати з платформою .NET.

Для розробки користувацького інтерфейсу програмного забезпечення також використаємо інтерфейс програмування додатків (API) Windows Forms, що є частиною .NET Framework. Бібліотека Windows Forms була створена для спрощення процесу розробки кросплатформного графічного інтерфейсу користувача.

Також для розробки програмного забезпечення, що оцінює розмір мобільних застосунків на Kotlin, обрано середовище розробки Visual Studio, оскільки воно є найбільш функціональним та офіційно підтримуваним компанією Microsoft. Також середовище розробки Visual Studio підтримує платформу .NET і надає широкі можливості з розробки програмного забезпечення для операційної системи Windows.

3.3.2 Діаграма компонентів програмного забезпечення

Діаграма компонентів розробляється для кращого розуміння структури проекту, візуалізації основних компонентів програмного забезпечення та взаємозв'язків між ними. Розроблена діаграма компонентів програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, представлена на рис. 3.8.

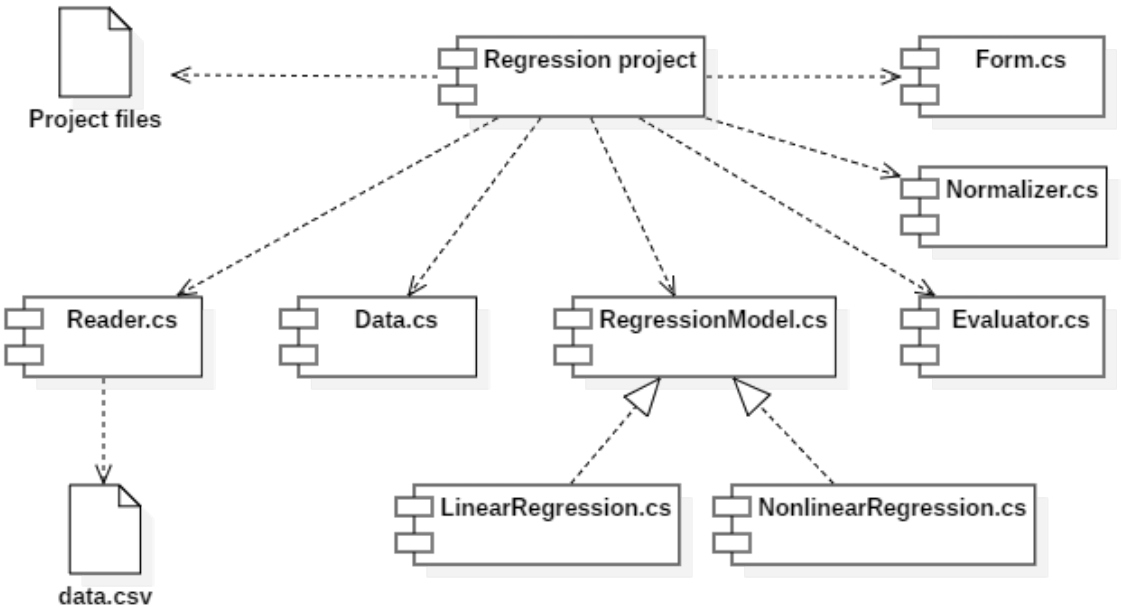

Рисунок 3.8 – Діаграма компонентів

Таким чином, у процесі робочого проектування була розроблена діаграма компонентів для розроблюваного програмного забезпечення.

3.3.3 Кодування та тестування програмного забезпечення

Для тестування розробленого програмного забезпечення використаємо метод еквівалентної розбивки. Для цього необхідно обрати варіанти використання для тестування, розбити їх на вхідну інформацію та певну кількість класів еквівалентності. Необхідно перевірити відповідність поведінки програми розробленому сценарію варіанту використання, це можна зробити із застосуванням методу «чорного ящика».

У даному випадку необхідно виконати тестування для варіанту використання «Введення вхідних даних». У табл. 3.11 наведено класи еквівалентності вхідних даних для варіанту використання «Введення вхідних даних».

Таблиця 3.11 – Класи еквівалентності вхідних даних для варіанту використання «Введення вхідних даних»

Вхідні дані	Правильні класи	Неправильні класи
Файл з розширенням типу .csv	Обраний файл звантажується коректно (1)	Файл має неправильне розширення (2) Вхідні дані мають неправильну структуру (3)

Результати тестування для правильних класів еквівалентності представлені в табл. 3.12.

Таблиця 3.12 – Тестування правильних класів еквівалентності для варіанту використання «Введення вхідних даних»

№ тесту	№ покритого класу	Вхідні дані	Вихідні данні
1	1	Файл з розширенням типу .csv	Обраний файл було успішно завантажено

Отже, із наведеної таблиці можна зробити висновок, що тестування правильних класів еквівалентності для варіанту використання «Введення вхідних даних» пройшло успішно.

У табл. 3.13 наведено результати тестування неправильних класів еквівалентності для варіанту використання «Введення вхідних даних».

Таблиця 3.13 – Тестування неправильних класів еквівалентності для варіанту використання «Введення вхідних даних»

№ тесту	№ покритого класу	Вхідні дані	Вихідні данні
1	2	Файл з розширенням типу .xlsx	Повідомлення про помилку
2	3	Файл з пошкодженою структурою даних	Повідомлення про помилку

Отже, із табл. 3.13 можна зробити висновок, що тестування неправильних класів еквівалентності варіанту використання «Введення вхідних даних» пройдено успішно.

3.3.4 Випробування програмного забезпечення

Для розробленого програмного забезпечення було виконано випробування згідно з програмою та методикою випробувань, яка наведена у додатку Д. Випробування полягало у послідовному введенні даних для тестування та проведенні аналізу отриманих результатів.

Випробування можливості введення вхідних даних у програму полягає у виконанні наступної послідовності дій:

- Натиснути на пункт меню «File».
- Натиснути на пункт меню «Load».
- Обрати шлях до файлу з вхідними даними.

Після виконання описаної послідовності дій файл з вхідними даними був успішно завантажений, а також здійснено розрахунки та виведено результати обчислень на екран, що відповідає очікуваному результату.

Результат випробування можливості введення вхідних даних у програму представлено на рис. 3.9.

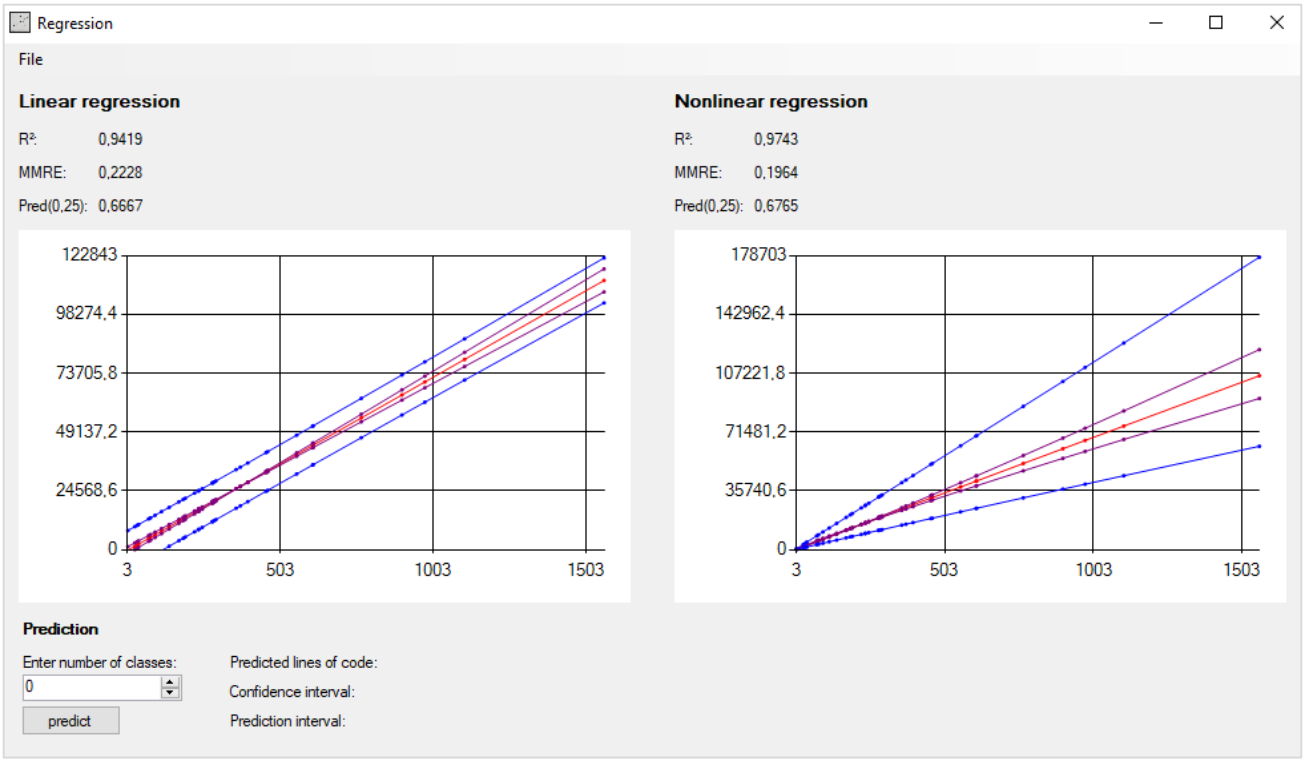

Рисунок 3.9 – Результати розрахунків після введення вхідних даних у програму

Випробування можливості розрахунку розміру мобільного застосунку полягає у виконанні наступної послідовності дій:

- Ввести кількість класів у числове поле, що знаходиться під надписом «Enter number of classes».
- Натиснути кнопку «predict».

Після виконання описаної послідовності дій буде виконане оцінювання розміру мобільного застосунку, результати обчислень будуть виведені на екран, що відповідає очікуваному результату.

Результат випробування можливості розрахунку розміру мобільного застосунку представлено на рис. 3.10.

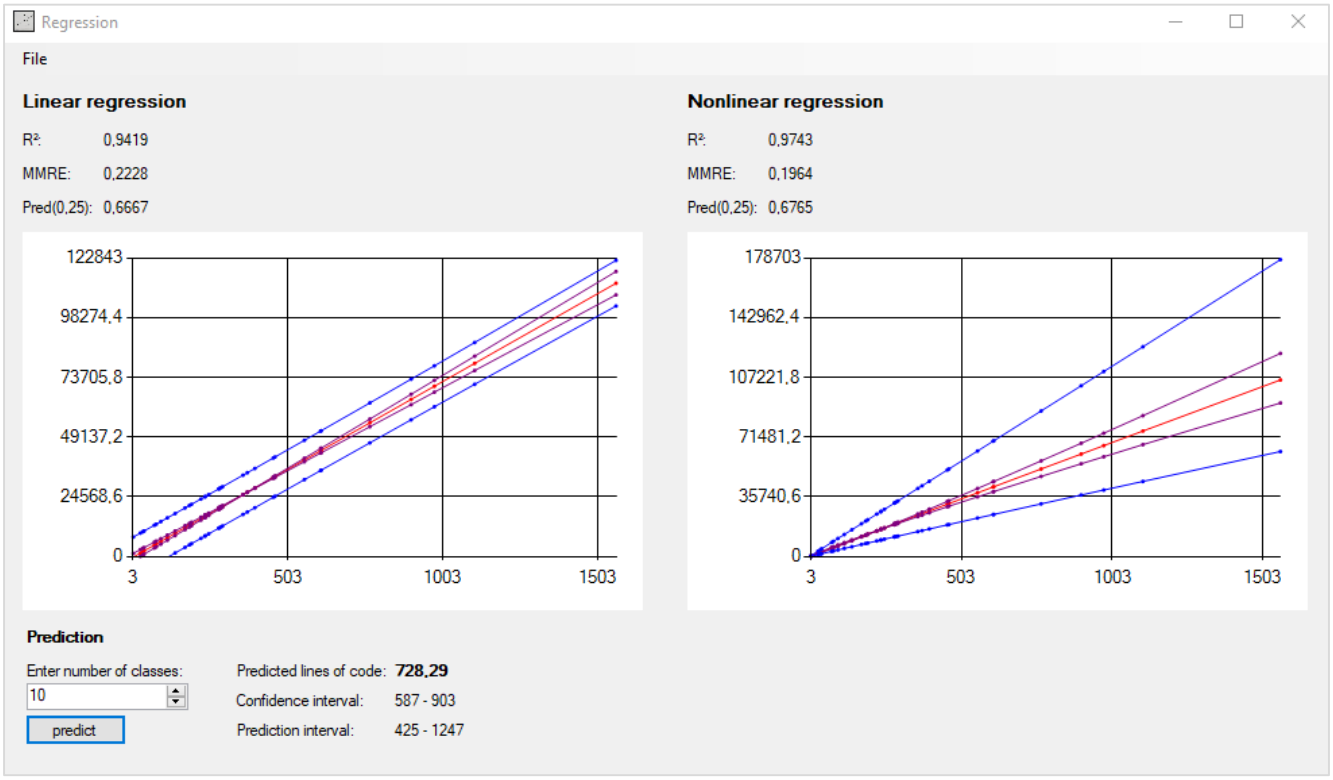

Рисунок 3.10 – Результат оцінювання розміру мобільного застосунку

Отже, у результаті проведення випробувань можна зробити висновок, що розроблене програмне забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, коректно виконує функції, що визначені у вимогах.

4 РЕЗУЛЬТАТ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОБІЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN

В процесі виконання кваліфікаційної роботи було розроблено програмне забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Розроблене програмне забезпечення виконує наступні перелічені функції:

- внесення даних про проекти мобільних застосунків (кількість рядків коду, кількість класів);
- виконання нормалізації внесених даних із застосуванням нормалізуючого перетворення на основі натурального логарифму;
- видалення викидів за наявності;
- побудова лінійної регресійної моделі із використанням нормалізованих даних;
- побудова довірчого інтервалу на основі нормалізованих даних;
- побудова інтервалу передбачення на основі нормалізованих даних;
- побудова нелінійної регресійної моделі на основі лінійної із застосуванням зворотного нормалізуючого перетворення;
- знаходження довірчого інтервалу нелінійної регресії на основі зворотного нормалізуючого перетворення;
- знаходження інтервалу передбачення нелінійної регресії на основі зворотного нормалізуючого перетворення;
- побудова лінійної регресійної моделі на основі вхідних емпіричних даних в припущенні про нормальність їх розподілу;
- оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

За допомогою побудови удосконаленої нелінійної регресійної моделі з використанням нормалізуючого перетворення на основі натурального логарифму була підвищена достовірність оцінювання розміру мобільних застосунків. Текст програми представлений у додатку Б.

Для розробленого програмного забезпечення були проведені випробування та зроблено висновок про відповідність програмного забезпечення усім вимогам, сформульованим у технічному завданні, яке наведене у додатку А. Програма та методика випробувань наведені у додатку Д.

Під час виконання роботи була розроблена інструкція користувача, що наведена у додатку Г, а також опис програми, що наведений у додатку В.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ МОБІЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN»

5.1 Вступ

Дана кваліфікаційна робота присвячена розробці програмного забезпечення для удосконалення нелінійної регресійної моделі, що дозволить ефективно прогнозувати розмір мобільних застосунків, які створюються мовою Kotlin. Дане програмне забезпечення може використовуватись при розробці мобільних застосунків на етапі планування програмного проекту для підвищення швидкості та достовірності оцінювання розміру мобільного застосунку.

Економічна ефективність – це величина, що виражена співвідношенням отриманих у процесі виробництва результатів, у даному випадку розробленого програмного забезпечення, до чинників і ресурсів, які були витрачені на виробництво.

Для обґрунтування доцільності розробки програмного забезпечення необхідно розрахувати собівартість розробленої системи, яка включає витрати на розробку, придбання матеріалів та технічних засобів та інші поточні витрати. Також необхідно визначити показники економії від впровадження розробленої системи з урахуванням витрат на експлуатацію. Економічна доцільність розробки визначається за допомогою показників річного економічного ефекту та строку окупності впровадженої системи.

У даному розділі наведено розрахунки наступних показників: вартість розробки програмного забезпечення, економічна ефективність у вигляді річного економічного ефекту, а також строк окупності розробленого програмного забезпечення.

5.2 Розрахунок витрат на створення та експлуатацію програмного забезпечення

Для розрахунку витрат на розробку програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, необхідно спочатку обчислити витрати на заробітну плату розробника, на амортизацію комп'ютера, що використовується при розробці програмного забезпечення, а також витрати на матеріали та засоби розробки.

Витрати на розробку програмного забезпечення спеціалістом складають 14000 грн. за місяць. Додаткова плата складає 10% від основної заробітної плати, а саме 1400 грн. Загальні витрати на оплату праці в місяць складають 15400 грн. Витрати на сучасний ноутбук складають 10000 грн (середня вартість ноутбука з наступними характеристиками: Intel Celeron N4000 / RAM 4 GB / SSD 128 GB / Intel UHD Graphics). Вартість електроенергії складає 1,44 грн за кіловат-годину. Також до витрат входять допоміжні матеріали, що перелічені в табл. 5.1.

Таблиця 5.1 – Розрахунок витрат на допоміжні матеріали

Найменування матеріалу	Вартість, грн.
Офісний папір	90
Заправка картриджу тонером для принтера	100
Доступ до мережі Internet	90
Непередбачені витрати	200
Сума	480

Вартість розробки програмного забезпечення розрахуємо за наступною формулою:

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

де $З_{зп}$ – витрати на оплату праці, що складаються з основної і додаткової заробітної плати (грн.);

$З_{сз}$ – єдиний соціальний внесок, що складає 38% від витрат на основну і додаткову заробітну плату (грн.);

$З_{зг}$ – загальногосподарські витрати, що складають 10% від витрат на оплату праці (грн.);

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

Т – загальна тривалість розробки (міс.);

$З_{м}$ – загальні витрати на допоміжні матеріали та засоби розробки (грн.);

За умови споживання потужності 0,3 Вт, тривалості розробки на місяць, а саме 22×8 = 176 годин, та при вартості кіловат-години електроенергії 1,44 грн., розрахуємо значення $З_{е}$:

$$З_{е} = 176 \times 1{,}44 \times 0{,}3 = 76{,}03 \text{ грн.}$$

Загальні витрати в місяць на розробку програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, наведено в табл. 5.2.

Таблиця 5.2 – Розрахунок витрат в місяць на розробку програмного забезпечення

Найменування витрат	Вартість, грн.
Загальні витрати на оплату праці	15400
Витрати на єдиний соціальний внесок	5852
Загальногосподарські витрати	1540
Загальні витрати на допоміжні матеріали та засоби розробки	480
Витрати на електроенергію	76,03

Оскільки за методом експертної оцінки на основі розробки аналогічних систем визначено, що загальна тривалість розробки Т складає 3 місяці, а кількість необхідних спеціалістів = 1 людина, то вартість розробки системи складає:

$$С_{пр} = (15400 + 5852 + 1540 + 76{,}03) \times 3 + 480 = 69084{,}09 \text{ грн.}$$

Розрахуємо амортизаційні відрахування на устаткування, у даному випадку ноутбук, що складають 60% від первісної вартості за один рік:

$$А_{об} = 10000 \times 0{,}6 = 6000 \text{ грн.}$$

Розрахуємо також витрати на матеріали за рік, що складають 5% від первісної вартості устаткування, а саме ноутбука:

$$В_{м} = 10000 \times 0{,}05 = 500 \text{ грн.}$$

Річний обсяг споживання електроенергії ноутбуком у годинах визначається наступним чином:

$$Ф_м = 253{,}3 \times Т_з,$$

де $Т_з$ – це середнє місячне завантаження устаткування, що складає приблизно 7 годин, а 253,3 – це середня кількість робочих днів у році.

Таким чином, розрахуємо значення річного обсягу роботи устаткування:

$$Ф_м = 253{,}3 \times 7 = 1773{,}1 \text{ год.}$$

Далі розрахуємо загальні витрати на електроенергію $З_е$ з урахуванням обсягу роботи устаткування за рік:

$$З_е = 1773{,}1 \times 0{,}3 \times 1{,}44 = 765{,}98 \text{ грн.}$$

Таким чином, загальні витрати на експлуатацію для ноутбука за один рік складатимуть:

$$З_{зр} = 6000 + 500 + 765{,}98 = 7265{,}98 \text{ грн.}$$

Отже, витрати на створення і експлуатацію програмного забезпечення у перший рік складатимуть:

$$З_{се} = 69084{,}09 + 7265{,}98 = 76350{,}07 \text{ грн.}$$

5.3 Економічна ефективність розробки та впровадження

Розроблене програмне забезпечення для оцінювання розміру мобільних застосунків на Kotlin дозволить підвищити якість та швидкість керування інформацією, а також знизить ризики допущення помилок, що є вагомим показником економічної ефективності.

Впровадження розробленого програмного забезпечення дозволить частково вивільнити робочий час працівників та значно підвищити продуктивність праці, що також впливає на економічну ефективність. Наслідком впровадження розробленої системи є зменшення ризику виникнення помилок при оцінюванні розміру мобільних застосунків, а також зменшення кількості помилкових рішень, прийнятих на основі отриманих неточних результатів оцінювання.

Розроблене програмне забезпечення дозволить підвищити швидкість та якість оцінювання розміру мобільних застосунків, а також значно покращити достовірність оцінювання, що впливає на поліпшення організації роботи та її своєчасність.

Оскільки впровадження розробленого програмного забезпечення дозволить вивільнити 0,5 робочого часу за експертним оцінюванням фахівців, з системою працює 1 спеціаліст, то умовно розроблена система вивільняє 0,5 працівників.

Розрахуємо оплату праці для 0,5 працівників за рік:

$$15000 \times 12 \times 0{,}5 = 90000 \text{ грн.}$$

Також необхідно розрахувати річний економічний ефект за наступною формулою:

$$\mathrm{E}_{\text{рік}} = \Delta\mathrm{C}_n - \mathrm{E}_n \times k,$$

де $\Delta\mathrm{C}_n$ – це кошти, які були вивільнені після впровадження розробленого програмного забезпечення, мінус витрати на експлуатацію, у даному випадку $\Delta\mathrm{C}_n = 90000 - 7265{,}98 = 82734{,}02$ грн.;

E_n – нормативний коефіцієнт ефективності, який дорівнює коефіцієнту амортизації 0,6;

k – одноразові капітальні витрати на впровадження розробленого програмного забезпечення, що складають 76350,07 грн.

Таким чином, маємо розрахунок річного економічного ефекту:

$$\mathrm{E}_{\text{рік}} = 82734{,}02 - 0{,}6 \times 76350{,}07 = 36923{,}978 \text{ грн.}$$

Строк окупності розробленого програмного забезпечення розрахуємо за наступною формулою:

$$T = \frac{k}{\Delta\mathrm{C}_n} = \frac{76350{,}07}{82734{,}02} \approx 0{,}92 \text{ року.}$$

Таким чином, строк окупності програмного забезпечення для оцінювання розміру мобільних застосунків на Kotlin складає приблизно 11 місяців.

5.4 Висновки

Отже, для програмного забезпечення для оцінювання розміру мобільних застосунків на Kotlin були здійснені розрахунки економічної ефективності від створення та впровадження, а також обчислено строк окупності розробленого програмного забезпечення після впровадження, який складає приблизно 11 місяців. Таким чином, можна зробити висновок, що розробка програмного забезпечення для оцінювання розміру мобільних застосунків на Kotlin є економічно вигідною.

6 ОХОРОНА ПРАЦІ

6.1 Вступ

Охорона праці – це сукупність організаційних та санітарно-гігієнічних заходів, що спрямовані на збереження працездатності, життя та здоров’я працівника у процесі трудової діяльності. Основні норми і вимоги до виробничої санітарії та техніки безпеки під час трудової діяльності, а також норми робочого часу та часу відпочинку регулюються законодавством про охорону праці. Основні положення щодо трудової діяльності визначаються законом України "Про охорону праці", який регулює відносини між працівником та власником підприємства, установи чи організації, де працівник здійснює трудову діяльність. Закон регулює такі питання як безпека працівника, дотримання санітарно-гігієнічних норм на підприємстві, а також загальні питання з охорони праці.

Законодавство з охорони праці базується на наступних принципах:

- пріоритетність здоров’я та життя людини, що здійснює трудову діяльність, а також наділення роботодавця повною відповідальністю за дотримання безпечних умов праці;
- дотримання допустимого рівня безпеки на робочих місцях шляхом виконання організаційних та технічних заходів, направлених на зниження впливу небезпечних та шкідливих факторів;
- забезпечення соціального захисту працівників, а також здійснення повного відшкодування шкоди потерпілим від небезпечних факторів на виробництві працівникам;
- встановлення вимог з питань охорони праці до підприємств, установ чи організацій, де здійснюється трудова діяльність;
- управління охорони праці в економічній сфері, а саме здійснення фінансування заходів, які стосуються питань охорони праці;

- здійснення заходів з інформування, а саме проведення професійної підготовки з питань охорони праці;

- забезпечення співробітництва роботодавців та працівників у процесі вирішення питань з охорони праці та прийняття рішень, пов’язаних з охороною праці;

- використання досвіду міжнародного співробітництва для покращення умов праці та підвищення рівня безпеки працівників.

Метою охорони праці є забезпечення умов, сприятливих для безпечної трудової діяльності. У процесі праці завжди існують певні небезпечні фактори, які є потенційним джерелом шкоди, тому важливо створювати такі умови, за яких вплив небезпечних та шкідливих факторів не перевищує гранично допустимих значень. Гранично допустиме значення шкідливого фактору – це такий рівень виробничого чинника, щоденний вплив якого не призводить до захворювань та зниження працездатності, а також не створює безпеку для здоров’я працівника. Безпека праці – це такі умови, за яких вплив небезпечних та шкідливих виробничих факторів максимально зменшується або виключається.

Безпека праці забезпечується за допомогою сукупності технічних та організаційних заходів, які спрямовані на усунення потенційних небезпек, що можуть створювати загрозу здоров’ю або життю працівника.

До технічних заходів входить наступні дії:

- підтримання чистоти приміщення, в якому здійснюється трудова діяльність, а також на робочих місцях працівників;

- дотримання санітарно-гігієнічних вимог до чистоти повітря у виробничому середовищі;

- забезпечення видалення пилу, парів, газів та інших шкідливих чинників з повітря у приміщенні, де здійснюється трудова діяльність, а також забезпечення вентиляції та кондиціювання;

- дотримання гранично допустимих рівнів шуму та різного виду випромінювання на робочих місцях.

Робоче місце – це частина території виробничої структури підприємства або організації, на якій постійно або тимчасово перебуває працівник під час трудової діяльності, зазвичай визначена трудовими нормами та обладнана певними засобами, необхідними для виконання трудових обов'язків. Робочі місця поділяють на постійні та тимчасові, зазвичай тимчасові робочі місця є частково або повністю автоматизованими, тому не потребують постійного перебування працівника.

Робоча зона – це певна визначена частина території, де знаходиться або постійне робоче місце працівника, або місце його тимчасового перебування. Робоча зона на відміну від робочого місця обмежена у висоту, зазвичай це обмеження у 2 метри над рівнем підлоги.

Для забезпечення продуктивної праці робітника необхідною є організація його робочого місця, а саме застосування заходів з планування, обладнання матеріалами та засобами праці, а також прийняття рішень щодо певного їх розміщення.

Отже, для забезпечення належних умов праці необхідно провести аналіз небезпечних та шкідливих факторів при роботі у офісі з екранними пристроями, а також визначити їх потенційний вплив на працездатність, здоров'я та життя працівника. Для забезпечення безпечних умов праці також необхідно визначити граничнодопустимі значення впливу виявлених чинників на робочих місцях.

6.2 Аналіз небезпечних і шкідливих факторів при роботі у офісному приміщенні з екранним пристроєм

Для аналізу небезпечних і шкідливих факторів необхідно спочатку розглянути умови праці при роботі в офісному приміщенні. Умови праці – це сукупність різних чинників, які мають вплив на працівників у процесі трудової діяльності. Серед основних чинників, що впливають на процес праці, можна виділити технології, обладнання, організацію робочих місць, організацію соціального захисту робітників, а також інші зовнішні фактори, що формують мікроклімат приміщення.

Фактори, що впливають на умови праці, умовно можна поділити на наступні:

- Виробничі: характеристики наявної техніки, ступінь автоматизованості праці, рівень організації робочих місць, організація режиму праці та відпочинку.

- Санітарно-гігієнічні: температура повітря в приміщенні, відносна вологість, ступінь забрудненості повітря, рівень шуму та вібрації, освітленість робочого місця.

- Психологічні: рівень комфорту на робочому місці, зручність користування технікою, зручність обслуговування, а також досконалість її конструкції;

- Фактори безпеки: рівень захисту працівників від виробничих травм, професійних захворювань, уражень струмом, а також різних видів випромінювання.

- Естетичні: рівень комфорту робочого середовища, привабливість його оформлення, а також комфортність рівня звуку.

- Соціальні: лояльність відносин у трудовому колективі, стиль керівництва, а також відповідність цілей організації інтересам працівника.

Для підвищення працездатності працівників необхідно створювати належні сприятливі умови праці, які забезпечать збереження працездатності та стану здоров’я робітників.

При роботі в офісі основними наслідками впливу небезпечних та шкідливих факторів є втома, стрес та поява травми. Надмірна інтенсивність та тривалість трудової діяльності людини призводить до втоми та тимчасового зниження працездатності. Розрізняють розумову, емоціональну та фізичну втому.

При роботі в офісі у результаті інтенсивної інтелектуальної діяльності може виникнути розумова втома, яка проявляється у вигляді зниження уваги, зниження розумової активності, уповільнення мислення, а також втрати інтересу до трудового процесу. Також можлива поява емоційної втоми у результаті впливу стресових факторів. Надмірне навантаження та високий рівень стресу на робочому місці протягом тривалого часу призводить до перевтоми, у наш час це є серйозною проблемою для багатьох фірм та організацій. Стрес може бути спричинений багатьма факторами, у тому числі високою інтенсивністю шуму у приміщенні.

Також вагомим фактором є низька якість освітленості приміщення та тривале використання комп'ютерів без перерви, що може призвести до напруги та втоми очей і у результаті поступового погіршення зору. Для зменшення впливу цього фактору необхідно забезпечити належну якість освітленості приміщення, а також здійснювати регулювання контрастності зображення на екрані комп'ютера. Для запобігання виникнення проблем з зором потрібно своєчасно робити перерви у процесі роботи з екранними пристроями.

При роботі у офісі також виникає небезпека ураження електричним струмом при неправильному розміщенні електричних дротів або при неправильному використанні подовжувачів в приміщенні офісу. Так, наприклад, якщо подовжувачі використовувати замість електричних розеток фіксованого розташування та підключають до них занадто велику кількість електроприладів, в результаті може виникнути перенапруження та небезпека ураження електричним струмом як наслідок. Також розповсюдженою є ситуація коли розмір і переріз шнура подовжувача не відповідає підключеному до нього електричному навантаженню по потужності.

Поширеною проблемою при роботі у офісі є погана якість повітря у приміщенні. Серед основних причин, що призводять до погіршення якості повітря можна виділити наступні:

- закрита, майже герметична організація споруди, як наслідок надходження недостатньої кількість зовнішнього повітря;
- занадто високе заповнення приміщення людьми;
- забруднення у результаті періодичного технічного обслуговування системи вентиляції;
- присутність у повітрі хімікатів від засобів очищення;
- поява цвілі у приміщенні;
- неправильна організація приміщення, наявність стін, що заважають вільній циркуляції повітря;
- рівень вологості, що не входить в гранично допустимий інтервал відносної вологості повітря;

Джерелом виникнення забруднювачів повітря у офісі часто є офісне обладнання. Оскільки якість повітря значно впливає на настрій та продуктивність праці робітників, а також може стати загрозою для здоров'я, наприклад призвести до астми, алергічних реакцій та інших респіраторних захворювань, то необхідно брати ці фактори до уваги та правильно планувати систему вентиляції в приміщенні.

Отже, для підвищення продуктивності праці та запобігання загроз для здоров'я працівників необхідно вживати заходів для покращення умов праці та зниження впливу небезпечних та шкідливих факторів, які є розповсюдженими при роботі в офісі.

6.3 Розрахунок системи штучного освітлення у офісному приміщені з екранними пристроями

При роботі у офісному приміщенні з екранними пристроями важливим фактором є якість освітлення. Низька якість освітлення може призвести до втоми очей працівників, головного болю, та у результаті поступового погіршення зору.

Частим наслідком низької якості освітлення у офісі є погіршення працездатності та загального самопочуття працівників. Тому важливо правильно здійснити планування освітлення у робочому приміщенні, а саме обрати тип та кількість лампочок у відповідності з нормами освітленості та розмістити їх відповідним чином.

Для різних типів приміщень в залежності від призначення існують різні показники якості освітлень. На основі цих норм з урахуванням площі приміщення потрібно розрахувати рівень освітленості.

За нормами робоче місце, обладнане комп’ютером має бути розташоване таким чином, щоб запобігати попадання прямого сонячного світла в очі працівника. Штучне освітлення у приміщенні має бути обладнане системою загального рівномірного освітлення.

Штучні джерела освітлення поділяються на загальні та місцеві. Зазвичай у офісі використовується комбіноване освітлення, тобто наявне загальне джерело світла, яке освітлює всі робочі місця, та місцеві джерела, які зосереджують свій світловий потік безпосередньо на окремих робочих місцях. Іноді також використовують тільки загальне джерело світла, але використання лише місцевого освітлення є неприпустимим, оскільки викликає необхідність частої адаптації зору та призводить до перенапруження очей працівників. Зазвичай у якості джерела світла в офісному приміщенні використовують люмінесцентні лампи.

Для розрахунку системи штучного освітлення приміщення використаємо метод на основі коефіцієнта світлового потоку, що вимірюється в люменах (Лм). Маємо наступні вхідні дані:

- довжина приміщення А = 5 м;
- ширина приміщення В = 6 м;
- висота приміщення Н = 3 м;
- коефіцієнт відображення стелі S_n = 70%;
- коефіцієнт відображення стін S_c = 50%;
- коефіцієнт відображення робочої поверхні S_p = 10%.

Для розрахунку світлового потоку F можна використати наступну формулу:

$$F = \frac{E_{min} \cdot k_з \cdot S \cdot z}{\eta},$$

де E_{min} – мінімальна освітленість приміщення, що визначається за таблицею, у даному випадку E_{min} = 150 Лк;

$k_з$ – коефіцієнт запасу, що залежить від виду лампи та ступеню забруднення приміщення;

S – площа приміщення, у даному випадку $S = 5 \cdot 6 = 30$ м^2;

z – коефіцієнт використання світлового потоку, що залежить від параметрів приміщення та використовуваних ламп;

η – значення, що визначається за таблицею коефіцієнтів використання і залежить від індексу приміщення i та коефіцієнтів відбиття поверхонь у приміщенні.

Індекс приміщення i розраховується за наступною формулою:

$$i = \frac{S}{h \cdot (A + B)},$$

де h – розрахункова висота підвісу, у даному випадку h = 2,6 м.

Підставимо параметри приміщення у формулу:

$$i = \frac{30}{2{,}6 \cdot (5 + 6)} = 1{,}05.$$

На основі розрахованого індексу приміщення за таблицею коефіцієнтів використання маємо $\eta = 0{,}36$.

Розрахуємо значення світлового потоку за формулою:

$$F = \frac{150 \cdot 1{,}5 \cdot 30 \cdot 1{,}15}{0{,}36} = 21562{,}5 \text{ Лм}.$$

Для освітлення приміщення офісу обрано люмінесцентні лампи із світловим потоком $F_л$ = 2500 Лм потужністю 60 Вт.

Розрахуємо необхідну для освітлення приміщення офісу кількість ламп за наступною формулою:

$$N = \frac{F}{F_л}.$$

Підставимо значення у формулу:

$$N = \frac{21562{,}5}{2500} = 9 \text{ шт}.$$

Таким чином, у результаті розрахунків можна зробити висновок, що для освітлення даного офісу потрібно використати 9 світильників з люмінесцентними лампами.

6.4 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів

Для зменшення впливу шкідливих і небезпечних виробничих чинників під час трудової діяльності з екранними пристроями у приміщенні офісу необхідно створити такі умови праці, які відповідатимуть нормам законодавства.

Закон України «Про охорону праці» встановлює вимоги до умов праці на робочому місці, до безпеки технологічних процесів та устаткування, до використовуваних засобів захисту, а також до санітарно-гігієнічних умов. Також встановлені вимоги до забезпечення освітлення приміщень, де здійснюється трудова діяльність.

При роботі з комп'ютером в приміщенні офісу особливо важливо правильно спланувати робоче місце, яке має відповідати виду та особливостям виконуваної роботи. Ергономічні властивості основних елементів робочого місця повинні буди достатньо зручними для працівника. Основні вимоги до ергономічних характеристик робочого місця при роботі з комп'ютером є наступними:

- ступні ніг розміщуються на підлозі або на підставці для ніг;
- стегна розміщуються в горизонтальній площині відносно підлоги;
- передпліччя розміщуються вертикально;
- лікті розміщуються під кутом 70-90° відносно вертикальної площини;
- зап'ястя зігнуті під кутом не більше 20° відносно горизонтальної площини;
- нахил голови приблизно 15-20° відносно вертикальної площини.

Робоче місце для працюючих з комп'ютером необхідно розташовувати таким чином, щоб до поля зору працівника не потрапляли освітлювальні прилади, вікна та поверхні, що мають властивість віддзеркалювання. Поверхня робочого столу має бути матовою. Для запобігання відблисків на екрані комп'ютера, екран монітора слід розміщувати таким чином, щоб світло від вікна падало збоку, бажано зліва.

Відстань від монітору комп'ютера до очей працівника повинен бути не менше 500-700 мм. Кут зору має бути в межах 10-40°. Клавіатура має бути розміщена на поверхні столу або на підставці на відстані 100-300 мм від краю столу. Кут нахилу клавіатури до горизонтальної поверхні має бути в межах 5-15°.

Крісло має забезпечувати працівнику зручні умови праці та фізіологічно комфортну позу в процесі трудової діяльності. Крісло має забезпечувати можливість регулювання висоти, а також куту нахилу спинки та висоти спинки.

У якості джерела штучного освітлення в приміщеннях офісу бажано використовувати люмінесцентні лампи.

З метою зменшення напруженості зору працівників потрібно забезпечити рівномірне розподілення яскравості робочої поверхні монітора та навколишнього простору приміщення.

Для запобігання накопичення пилу у приміщенні та уникання погіршення стану повітря необхідно щодня проводити вологе прибирання та регулярно провітрювати офіс протягом робочих днів.

Для створення комфортних умов та підвищення працездатності працівників необхідно забезпечити температуру навколишнього середовища в межах 18-22°С та відносну вологість повітря близько 55%.

Для запобігання небезпечних ситуацій, пов’язаних з ураженням струмом, необхідно перевіряти цілісність кабелів живлення та місця їх підключення. Бажано перед початком роботи візуально перевіряти цілісність корпусів системного блоку та монітору.

Рекомендовано при роботі з екранними пристроями періодично робити перерви для відпочинку з метою зменшення негативного впливу різних виробничих факторів ризику, пов’язаних з роботою з комп’ютером. При роботі з текстом бажано встановлювати великий шрифт та не допускати занадто довге фокусування очей на моніторі.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

7.1 Забруднення навколишнього середовища в процесі використання комп'ютера

Основним завданням охорони навколишнього середовища є вирішення питання щодо раціонального використання і відтворення природних ресурсів, а також прагнення до збереження екологічної безпеки за допомогою зменшення впливу господарської та виробничої діяльності на навколишнє середовище.

Під забрудненням навколишнього середовища розуміють сукупність дій, що призводять до порушення природних процесів та до негативних змін в структурі компонентів екосистеми. В результаті таких змін можуть бути порушені процеси обміну речовин, що призводить до зниження продуктивності та руйнування екосистеми.

Забруднення можна поділити за типом речовин, що їх спричинили:

- фізичні забруднення: шумове та вібраційне забруднення, зміни теплових, та радіаційних полів;
- механічні забруднення: пил та різні тверді частки, наприклад викинуті як непридатні;
- хімічні забруднення: хімічні елементи та їх сполуки штучного походження, фтористі з'єднання, важкі метали;
- біологічні забруднення: організми, що з'являються у результаті діяльності людей, наприклад нові віруси, бактеріологічна зброя, а також надмірне розмноження рослин або тварин, що були переселені з одного середовища в інше;

Охорона навколишнього середовища в Україні регулюється Міністерством екології і природних ресурсів, а саме законом України «Про охорону навколишнього природного середовища». Даний закон визначає організацію економічної, правової та соціальної сфер охорони природи навколишнього природного середовища і націлений на інтереси нинішнього та майбутніх поколінь.

У даному законі також визначено перелік природних об’єктів, які потребують особливої охорони. Також закон передбачає розробку нових виробничих і утилізаційних технологій, що допоможуть зменшити антропогенне навантаження на довкілля. Жорсткі вимоги до виробничої діяльності та використовуваних матеріалів є необхідними для зменшення негативного людського впливу на навколишнє середовище. Не менш важливим джерелом екологічного права в Україні є Конституція, яка закріплює основи екологічного права та має вищу юридичну силу.

На сьогоднішній день інформаційні технології є невід’ємною частиною процесу виробництва та більшості видів людської діяльності, тому неможна не брати до уваги вплив комп’ютерів на навколишнє середовище на кожній стадії їхнього використання – виробництво, експлуатація і утилізація після закінчення терміну служби.

Вимоги до виробництва і складу матеріалів, що використовуються в конструкціях комп'ютерів, визначаються сучасними екологічними стандартами. Використовувані матеріали не повинні містити хлоридів і бромідів, як засоби захисту від згорання, а також фреон, який руйнує озоновий шар. Встановлено також обмеження до вмісту кадмію у світлочутливому шарі екрана монітора, а також до вмісту ртуті у батарейках. Існують достатньо чіткі вимоги відносно видів пластмаси та лаків для покриття, що використовуються при виробництві комп’ютерів. Нажаль відмовитися від використання свинцю при виробництві наразі неможливо. Поверхня кнопок не повинна містити такі матеріали як нікель, хром та інші матеріали, що можуть спричинити алергічну реакцію у користувача.

Залишки кабелів, металу, різних елементів електронних схем, а також частини комплектуючих та застарілу техніку що вийшла з строку експлуатації збирають в спеціальні контейнери та утилізують у спеціалізованих відведених для цього місцях.

Сучасні міжнародні стандарти також включають вимоги зниженого енергоспоживання, а також обмежують припустимі рівні потужності в режимі споживання.

Таким чином, згідно з законом України «Про охорону навколишнього природного середовища» персональні комп'ютери у процесі використання не є джерелом будь-яких шкідливих речовин, які забруднюють навколишнє середовище, але під час виробництва важливо дотримуватись вимог, визначених у стандартах виробництва і складу матеріалів, а також правильно здійснювати утилізацію після закінчення строку експлуатації.

7.2 Розробка заходів щодо зменшення забруднення навколишнього середовища

На сьогоднішній день проблема забруднення навколишнього середовища у процесі експлуатації комп'ютерів виробниками комп'ютерних технологій є досить актуальною. Для вирішення цього питання було створено так звані «Зелені технології», які представлені комплексом екологічно нешкідливих, або менш шкідливих порівняно з традиційними способів виробництва.

Дана сукупність технологій реалізована в економічній, екологічній, технологічній та інноваційній сферах і направлена на вирішення таких питань як переробка відходів та використання альтернативних джерел електроенергії.

Зелені технології дозволяють вирішувати наступні завдання:

- Сприяння сталому розвитку із запобіганням виснаження ресурсів.
- Виробництво товарів, що можуть бути перероблені, відновлені або повторно використані.
- Прагнення до зменшення забруднення довкілля та підвищення ефективності використання ресурсів при виробництві.
- Застосування інновацій, що дозволять замінити застарілі способи виробництва, які завдають шкоди навколишньому середовищу.
- Сприяння економічному розвитку, створенню нових технологій та товарів.
- Економія енергії у різних галузях виробництва, сільського господарства, культури та урбаністики.
- Послаблення кліматичних змін та зниження суспільної вразливості до них.

На сьогоднішній день все більше компаній звертаються до інтелектуальних автономних систем, які дозволяють здійснювати віддалений моніторинг та управління. Потенціал цієї технології полягає у можливості скорочення витрат та підвищенні рівня безпечності.

Віддалена робота стає все більш розповсюдженою, у результаті чого відбувся перехід до систем хмарних обчислень для зберігання файлів та програм, що допомогло підприємствам зменшити споживання енергії. Використання систем хмарних обчислень стало не лише зручним, а й екологічним рішенням.

Також потрібно відзначити, що технологія 5G відіграватиме все більш важливу роль у забезпеченні сталого розвитку та збереженні природних ресурсів завдяки різним сценаріям їх використання. За оцінками експертів подальше використання 5G скоротить глобальні викиди парникових газів на 1,7 мільярдів тон.

До переваг впровадження зелених технологій відносяться, насамперед, поліпшення стану довкілля та здоров'я людей, збереження ресурсів, підвищення ефективності виробництва, та у результаті підвищення конкурентоспроможності продукції, що випускається.

Сталий розвиток – це такий розвиток суспільства, при якому поліпшуються умови життя людей, а вплив на довкілля залишається у межах господарської ємності біосфери, отже не руйнується природна основа функціонування людства. При сталому розвитку задоволення потреб здійснюється без завдавання шкоди майбутнім поколінням. Концепція сталого розвитку є передумовою довгострокового прогресу людства, супроводжуваного збільшенням капіталу та поліпшенням екологічних умов.

Концепцію сталого розвитку можна визначити у вигляді п'яти головних принципів:

- Людство має прагнути до сталого розвитку, розглядаючи його з довготривалої точки зору, але необхідно брати до уваги потреби не лише теперішніх поколінь, але й надавати можливість задовольняти свої потреби майбутнім поколінням.

- Необхідно брати до уваги, що обмеження в галузі експлуатації природних ресурсів є відносними і базуються на сучасному рівні техніки, соціальній організації, а також на здатності біосфери до самовідновлення.

- Задоволення елементарних потреб всіх людей є обов'язковим, всім необхідно надавати можливість забезпечити собі благополучне життя. Це є необхідною умовою сталого і довготривалого розвитку.

- Необхідно урівноважити життєвий стан тих, хто користується надмірними грошовими і матеріальними засобами, з екологічними можливостями планети. Особливо це стосується надмірного використання енергії.

- Необхідно узгодити темпи і розміри росту населення з виробничим потенціалом загальної екосистеми планети.

Технічні директори компаній-виробників комп'ютерних технологій повинні прагнути до того, щоб впровадження принципів сталого розвитку та етичних норм у їх виробництво стало основною частиною їх стратегій. Вони мають відігравати важливу роль у відстоюванні бізнес-моделей сталого розвитку. Це важлива можливість використати технології для підтримки досліджень, зусиль з пониження наслідків змін клімату та сприятливих для клімату рішень. Крім цього екологічні IT компанії також можуть допомогти скоротити витрати там, де використовують інтелектуальні технології, а переробка матеріалів може сприяти розвитку економіки замкнутого циклу.

Не менш вагомим фактором є надмірне споживання електроенергії виробниками комп'ютерних технологій. Так, операторам дата-центрів доводиться не лише забезпечувати роботу десятків серверів, а й стежити їх охолодженням, що потребує надмірних витрат електроенергії.

Часто при цьому сервери виявляються завантаженими лише 10-20%. Використання спеціального програмного забезпечення з області «зелених технологій» дозволяє значно ефективніше розподілити навантаження та відмовитися від використання надмірної кількості комп'ютерів. Проте на шляху запровадження зелених технологій існують перешкоди.

Одна з основних причин того що компанії не наважуються впроваджувати зелені технології полягає в наступному. Вони вважають, що це буде дорожче, ніж поточна технологія, що вже використовується в їх діяльності.

Компанії, які витрачають додаткові кошти на забезпечення екологічної чистоти, зустрічаються не часто. Особливо якщо необхідно вжити заходів, які вимагають великого терміну окупності або не окупаються зовсім. Серед організацій, які зважилися на це, можна виділити Google, що розпочала реалізацію масштабної ініціативи з використання сонячної енергії для забезпечення незліченої кількості своїх серверів електроенергією.

Часто компанії, що розглядають можливість впровадження зелених технологій, орієнтуються на дуже тривалі терміни окупності. У згаданому випадку з Google очікується, що термін окупності складе 7,5 років навіть при значних податкових пільгах і субсидіях на розвиток електромереж живлення. І в кращому випадку Google зможе задовольнити за рахунок сонячної енергії лише 30% своїх потреб у електроенергії.

Насправді, хоча екологічні рішення справді асоціюють із високими початковими витратами, вони набагато більш ефективні та протягом усього терміну експлуатації. Часто менеджери, директори та інші зацікавлені сторони скептично ставляться до переваг зелених технологій і тому не наважуються на застосування «зелених інновації» у бізнесі.

Негативні наслідки впровадження «зелених технологій» характерні лише на початковому етапі, і коли він мине, зелені технології дозволять скоротити витрати та стануть каталізатором інновацій і джерелом нових ринкових можливостей.

Можна зробити висновок, що на даний момент впровадження зелених технологій набирає популярність досить повільно, і меншою мірою сприяє зменшенню витрат, проте дозволить споживачам та виробникам робити свій внесок у захист навколишнього середовища. Але з часом ймовірно, що вибір на користь екологічних рішень не буде підсвідомо прирівнюватися до певних обмежень або зниження рівню комфорту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було удосконалено нелінійну регресійну модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin. Поставлені завдання було виконано у повному обсязі:

- проведено аналіз існуючих методів та моделей для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin;
- здійснено обґрунтування актуальності удосконалення регресійної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin;
- зібрані вихідні дані з завершених проектів мобільних застосунків, реалізованих мовою Kotlin, які були використані для побудови удосконаленої регресійної моделі;
- здійснено удосконалення регресійної моделі шляхом переходу від лінійної регресійної моделі до нелінійної, із застосуванням зворотного нормалізуючого перетворення на основі натурального логарифму;
- розроблено програмне забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

В результаті виконання роботи нелінійна регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, була удосконалена за критерієм рівня прогнозування на 0,0098, за значенням середньої величини відносної похибки на 0,0264 та за значенням коефіцієнту детермінації на 0,0324.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Котов, С.Л. Информационно-аналитическая система оценивания трудозатрат и стоимости создания программных средств [Текст] / С.Л. Котов, А.А. Демирский // Программные продукты и системы. – 2017. – Т. 30, № 3. – С. 469-474.

2. Кузнецова, А.С. Регресійна модель для оцінювання розміру мобільних застосунків, що створені мовою Kotlin / А.С. Кузнецова, І.В. Устенко //Матеріали IV Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць (30 листопада 2021 року) / Під редакцією Г.О. Райко. – Херсон: Видавництво ФОП Вишемирський В. С., 2021. – С. 40-42.

3. Prykhodko N. V. The non-linear regression model to estimate the software size of open source java-based systems / N. V. Prykhodko, S. B. Prykhodko // Радіоелектроніка, інформатика, управління. - 2018. - № 3. - С. 158-166.

4. Prykhodko S.B. Constructing the non-linear regression equation to estimate the software size of open source PHP-based information systems / Prykhodko S.B., N. V. Prykhodko, A. V. Spinov // Проблеми інформаційних технологій, 2018 – С. 118-125.

5. The Kotlin Programming Language. [Електронний ресурс] / Режим доступу: https://github.com/JetBrains/kotlin/ – Назва з екрану.

6. Kotlin is now Google's preferred language for Android app development [Електронний ресурс] / Режим доступу: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/ – Назва з екрану.

7. Титов, А.И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ [Текст] / А.И. Титов // Интеллектуальные технологии на транспорте. – 2016. – № 1. – С. 31-38.

8. Титов, А.И. Выбор показателей для сравнения метрик размера проекта [Текст] / А.И. Титов // Перспективы развития информационных технологий. – 2016. – № 29. – С. 17-18.

9. Домакур, О.В. Анализ методик оценки трудоемкости разработки программного обеспечения / А.В. Будник, О.В. Домакур, Е.С. Романова, Т.Л. Труханович // Веснік сувязі. — 2020. — № 2 (162). — С. 36–41.

10. Приходько, С.Б. Побудова нелінійних регресійних рівнянь на основі багатовимірних нормалізуючих перетворень [Текст] / С.Б. Приходько, Н. В. Приходько // Вісник Харківського національного університету імені В.Н. Каразіна, серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління», 2018. – № 3 (39). – С. 61-68.

11. Приходько, С.Б. Інтервальне оцінювання статистичних моментів негаусівських випадкових величин на основі нормалізуючих перетворень [Текст] / С.Б. Приходько // Математичне моделювання: науковий журнал. – Дніпродзержинськ: ДДТУ, 2011. – №1 (24). – С. 9-13.

12. Тютюнников Н.Н. Оценка размера создаваемого программного средства с использованием функциональных точек [Текст] // Перспективы развития информационных технологий, 2014. - №18. - 207 с.

13. Malathi, S. Analysis of size metrics and effort performance criterion in software cost estimation. Indian Journal of Computer Science and Engineering, 2012 – pp. 24-31.

14. Гмурман, В.Е. Теория вероятностей и математическая статистика. / Гмурман, В.Е. – М.: Высшее образование, 2010. – С. 176-177.

15. Prykhodko S.B. Developing the software defect prediction models using regression analysis based on normalizing transformations / S.B. Prykhodko // in “Modern problems in testing of the applied software” (PTTAS-2016), Abstracts of the Research and Practice Seminar, Poltava, Ukraine, May 25-26, 2016, pp. 6-7.

16. Chatterjee, Samprit. Handbook of Regression Analysis [Text] / Samprit Chatterjee, Jeffrey S. Somonoff. Wiley, 2012. – 240 p.

17. Приходько С.Б. Методичні вказівки до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на ЕОМ" [Текст] / С.Б. Приходько – Миколаїв: НУК, 2005. – 52 с.

18. Вентцель, Е.С. Теория вероятностей: Учеб. для вузов [Текст] / Е.С. Вентцель. – М.: Высш. шк., 1999. – 576 с.

19. Лемешко Б. Ю. Критерии согласия типа хи-квадрат при проверке нормальности / Б. Ю. Лемешко // Измерительная техника. - 2015. - № 6. - С. 3-9.

20. Prykhodko S. Multivariate outlier detection technique based on normalizing transformations for non-Gaussian data / S. Prykhodko, N. Prykhodko, L. Makarova, K. Pugachenko // «Інформаційні технології та комп`ютерне моделювання»: матеріали статей Міжнародної науково-практичної конференції, м. Івано-Франківськ, 15-20 травня 2017 року. – ІваноФранківськ: п. Голіней О.М., 2017. – С. 170-174.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ

Вступ

Для полегшення процесу планування проектів з розробки мобільних застосунків мовою Kotlin необхідно створити програмне забезпечення, яке дозволить отримати прогнозований розмір мобільного застосунку на основі удосконаленої математичної моделі.

Назва програмного забезпечення: "Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin".

1 Підстави для розробки

Підставою для розробки програмного забезпечення є завдання на кваліфікаційну роботу "Регресійна модель для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, та розробка програмного забезпечення для її реалізації".

2 Призначення розробки

2.1 Функціональне призначення

Функціональнім призначенням програмного забезпечення є побудова та удосконалення математичної моделі для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, а також виведення результатів розрахунків на екран.

2.2 Експлуатаційне призначення

Програмне забезпечення може використовуватись розробниками для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.2 Вимоги до складу виконуваних функцій

Програмне забезпечення має надавати можливість виконувати наступний перелік функцій:

- внесення вхідних даних про проекти у вигляді кількості рядків коду та кількості класів;
- застосування до внесених даних нормалізуючого перетворення на основі натурального логарифму;
- побудова лінійної регресійної моделі;
- побудова довірчого інтервалу та інтервалу передбачення для лінійної регресії;
- побудова нелінійної регресійної моделі з використанням зворотного нормалізуючого перетворення;
- розрахунок довірчого інтервалу та інтервалу передбачення для нелінійної регресії за допомогою зворотного нормалізуючого перетворення;
- розрахунок показників якості для лінійної та нелінійної регресійних моделей;
- розрахунок оцінки розміру мобільного застосунку;
- виведення отриманих розрахунків на екран.

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані мають бути організовані у вигляді файлів з розширенням .csv та вводяться за допомогою вибору шляху до відповідного файлу у вікні операційної системи.

Вихідні дані, а саме довірчі інтервали, інтервали передбачення, метрики якості та результат розрахунку оцінки розміру мобільного застосунку мають виводитись на екран користувача.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного функціонування програмного забезпечення

Стійке функціонування програмного забезпечення повинно забезпечуватись шляхом організації безперебійного живлення для комп’ютера, а також за умови стабільного функціонування операційної системи.

У випадку виникненні збоїв в роботі програмного забезпечення, для відновлення нормальної роботи необхідно здійснити перезавантаження операційної системи та самої програми.

3.2.2 Вимоги до контролю вхідної та вихідної інформації

Програмне забезпечення має перевіряти вхідні дані на коректність. У випадку завантаження користувачем файлу неправильного типу, або некоректної структури, користувач має отримати повідомлення про помилку.

3.2.3 Вимоги до часу відновлення після відмови

Для відновлення після відмови потрібен час, який складається з часу перезавантаження операційної системи, перезавантаження самої програми, та повторного введення вхідних даних.

3.3 Вимоги до умов експлуатації

Вимоги до умов експлуатації програмного забезпечення відповідають загальним вимогам до умов експлуатації персонального комп'ютера, а саме:

- температура повітря у приміщенні: від +10°С до +35°С;
- відносна вологість повітря: 40-60%;
- атмосферний тиск: 630-800 мм ртутного стовпа;
- запиленість повітря: не більше ніж 0,75 мг/м3;

Користувач програмного продукту повинен мати навички роботи з персональним комп'ютером та володіти основними знаннями предметної галузі.

3.4 Вимоги до складу і параметрів технічних засобів

Для роботи з програмним забезпеченням необхідною умовою є наявність персонального комп'ютера з наступними мінімальними системними характеристиками:

- процесор: 32-розрядний процесор (x86), 1ГГц;
- обсяг оперативної пам‘яті: 2 Гб;
- обсяг вільного дискового простору: 500 Мб.
- наявність пристроїв вводу та виводу: монітор, клавіатура, миша;

3.5 Вимоги до інформаційної і програмної сумісності

Для роботи з програмним забезпеченням необхідно використовувати ліцензійну версію операційної системи Windows.

4 Вимоги до програмної документації

Програмна документація повинна складатись з наступних документів:

технічне завдання; текст програмних модулів; опис програмного забезпечення; програма та методика випробовування; інструкція для користувача програмного забезпечення;

5 Техніко - економічні показники

Техніко-економічні показники розробки та впровадження програмного забезпечення були розраховані та представлені в 5 розділі кваліфікаційної роботи.

6 Стадії та етапи розробки

Стадії та етапи розробки програмного забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin, а також терміни виконання, наведені у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадія розробки	Етап розробки	Зміст роботи	Термін виконання
1	2	3	4
Технічне завдання	Розробка і затвердження технічного завдання	Постановка задачі та обґрунтування необхідності проведення досліджень, визначення вимог до програмного забезпечення, розробка технічного завдання та його затвердження	Початок: 15.09.21 Закінчення: 30.09.21

Продовження таблиці А.1

1	2	3	4
Ескізний проект	Розробка і затвердження ескізного проекту	Виявлення основних вимог, розробка структури та алгоритму поведінки майбутньої системи, розробка ескізного проекту та його затвердження	Початок: 30.09.21 Закінчення: 15.10.21
Технічний проект	Розробка і затвердження технічного проекту	Уточнення алгоритму рішення задачі та структури програмного забезпечення, розробка технічного проекту та його затвердження	Початок: 15.10.21 Закінчення: 30.10.21
Робочий проект	Розробка і затвердження робочого проекту	Розробка програмного продукту та його документації, розробка інструкції користувача, розробка робочого проекту та його затвердження	Початок: 30.10.21 Закінчення: 20.11.21

7 Порядок контролю та приймання

Контроль та приймання програмного забезпечення має здійснюватися замовником або його представником у відповідності з програмою та методикою випробувань. Кожен етап проектування та розробки програмного забезпечення повинен відповідати вимогам, що зазначені у технічному завданні, а також реалізовуватись у зазначені строки.

У результаті проведення приймання необхідно скласти акт, який мають підписати замовник та розробник, або її представники. У випадку виявлення помилок розробник зобов'язаний їх виправити протягом не більше ніж 1-го місяця та провести повторне приймання програмного забезпечення.

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

Далі наведено лістинг коду програми

Form.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Regression
{
    public partial class Form : System.Windows.Forms.Form
    {
        private Reader reader;
        private NonlinearRegression nonlinearRegression;

        public Form()
        {
            InitializeComponent();
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Stream stream = null;
            OpenFileDialog fileDialog = new OpenFileDialog();

            fileDialog.InitialDirectory = "c:\\";
            fileDialog.Filter = "csv files (*.csv)|*.csv|All files (*.*)|*.*";
            fileDialog.FilterIndex = 1;
            fileDialog.RestoreDirectory = true;

            if (fileDialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    if ((stream = fileDialog.OpenFile()) != null)
                    {
                        using (stream)
                        {
                            reader = null;
                            reader = new Reader(stream);
                            string[] header = reader.get_Header();
                            List<string> lX = new List<string>();
                            List<string> lY = new List<string>();
                            for (int i = 0; i < header.Length; i++)
                            {
                                lX.Add(header[i]); lY.Add(header[i]);
                            }
                            // Close the stream
                            stream.Close();
                        }
                    }
```

```
                    if (reader != null)
                    {
                        //Get input data for building linear regression
                        double[][] inputData = reader.get_Data();

                        //Build linear regression
                        LinearRegression linearRegression = new LinearRegression(new
Data(inputData));
                        //Show regression on the chart
                        Plot plot1 = new Plot(linearRegression, chartLinear);

                        //Show evaluations for linear regression
                        R2_lin_value.Text = getFormattedR2Evaluation(linearRegression);
                        mmre_lin_value.Text = getFormattedMMREEvaluation(linearRegression);
                        pred25_lin_value.Text = getFormattedPredEvaluation(linearRegression);



                        //Build nonlinear regression using linear regression model with
normalized data
                        //Normalize data
                        Data normalData = Normalizer.normalizeAndDeleteErrors(inputData);
                        nonlinearRegression = new NonlinearRegression(new
LinearRegression(normalData));
                        //Show regression on the chart
                        Plot plot2 = new Plot(nonlinearRegression, chartNonlinear);

                        //Show evaluations for nonlinear regression
                        r2_nonlin_value.Text = getFormattedR2Evaluation(nonlinearRegression);
                        mmre_nonlin_value.Text =
getFormattedMMREEvaluation(nonlinearRegression);
                        pred25_nonlin_value.Text =
getFormattedPredEvaluation(nonlinearRegression);

                        classes_numeric.Enabled = true;
                        predict_button.Enabled = true;
                    }
                    else
                    {
                        MessageBox.Show("Error, no data to plot! Please load csv file");
                        return;
                    }
                }
                catch (Exception err)
                {
                    //Inform the user if we can't read the file
                    MessageBox.Show("Selected file has incorrect structure or wrong
extension");
                    clearPrediction();
                    //MessageBox.Show(err.Message);
                }
            }
        }

        private string getFormattedR2Evaluation(RegressionModel regression)
        {
            return Math.Round(Evaluator.getR2(regression), 4).ToString();
        }

        private string getFormattedMMREEvaluation(RegressionModel regression)
        {
            return Math.Round(Evaluator.getMMRE(regression), 4).ToString();
        }
```

```
        private string getFormattedPredEvaluation(RegressionModel regression)
        {
            return Math.Round(Evaluator.getPred25(regression), 4).ToString();
        }

        private void predict_button_Click(object sender, EventArgs e)
        {

            //Get entered number of classes
            int classes = (int)classes_numeric.Value;

            if(classes <= 0)
            {
                MessageBox.Show("The number of classes must be greater than 0");
                clearPrediction();
                return;
            }

            lines_text.Text = getFormattedLinesOfCode(classes);
            prediction_delta_label.Text = getFormattedPredIntervalDelta(classes);
            confidence_delta_label.Text = getFormattedConfIntervalDelta(classes);
        }

        private string getFormattedPredIntervalDelta(int classes)
        {
            return getFormattedIntervalDelta(nonlinearRegresion.getPredIntervDelta(classes));
        }

        private string getFormattedConfIntervalDelta(int classes)
        {
            return getFormattedIntervalDelta(nonlinearRegression.getConfIntervDelta(classes));
        }

        private string getFormattedIntervalDelta(double interval)
        {
            return "± " + getRoundedNumber(interval).ToString();
        }

        private string getFormattedLinesOfCode(double classes)
        {
            return getRoundedNumber(nonlinearRegression.getYCalculated(classes));
        }

        private string getRoundedNumber(double number)
        {
            return Math.Round(number, 2).ToString();
        }

        private void clearPrediction()
        {
            lines_text.Text = "";
            prediction_delta_label.Text = "";
            confidence_delta_label.Text = "";
        }

    }
}
```

Reader.cs

```
using System;
using System.IO;

namespace Regression
{
    class Reader {
```

```
        private string[] header;
        private double[][] data;
        private int nLines;
        private int nColumns;

        public Reader(Stream stream)
        {
            string line;

            //read the file line by line
            StreamReader streamReader = new StreamReader(stream);
            line = streamReader.ReadLine();
            header = line.Split(',');
            nColumns = header.Length;
            nLines = 0;
            while ((line = streamReader.ReadLine()) != null)
            {
                if (line.Length > 0) nLines++;
            }

            //read the numerical data from file in an array
            data = new double[nLines][];
            streamReader.BaseStream.Seek(0, 0);
            streamReader.ReadLine();
            for (int i = 0; i < nLines; i++)
            {
                data[i] = new double[nColumns];
                line = streamReader.ReadLine();
                string[] dataPair = line.Split(',');
                for (int j = 0; j < nColumns; j++)
                    data[i][j] = double.Parse(dataPair[j]);
            }
            streamReader.Close();
        }

        //functions used for retrieving the data
        public int get_nLines()
        {
            return nLines;
        }

        public int get_nColumns()
        {
            return nColumns;
        }

        public double[][] get_Data() => data;

        public string[] get_Header()
        {
            return header;
        }
    }
}
```

Normalizer.cs

```
using System;

namespace Regression
{
    static class Normalizer
    {
        public static double normalize(double numb)
        {
```

```
    return Math.Log(numb);
}

public static double reverse(double numb)
{
    return Math.Round(Math.Exp(numb));
}

public static Data normalizeAndDeleteErrors(double[][] inputData)
{
    double[][] nData = normalizeData(inputData);
    while (hasErrors(nData))
    {
        nData = deleteErrors(nData);
    }

    return new Data(nData);
}

public static double[][] normalizeData(double[][] data)
{
    double[][] normalizedData = new double[data.Length][];

    for (int i = 0; i < data.Length; i++)
    {
        normalizedData[i] = new double[2];
        normalizedData[i][0] = normalize(data[i][0]);
        normalizedData[i][1] = normalize(data[i][1]);
    }

    return normalizedData;
}

public static double[][] reverseData(double[][] data)
{
    double[][] reversedData = new double[data.Length][];

    for (int i = 0; i < data.Length; i++)
    {
        reversedData[i] = new double[2];
        reversedData[i][0] = reverse(data[i][0]);
        reversedData[i][1] = reverse(data[i][1]);
    }

    return reversedData;
}

public static bool hasErrors(double[][] data)
{
    bool flag = false;
    LinearRegression regr = new LinearRegression(new Data(data));
    double[][] clearData = new double[data.Length][];
    double[][] posPredInterval = regr.getPosPredInterval();
    double[][] negPredInterval = regr.getNegPredInterval();

    for (int i = 0; i < data.Length; i++)
    {
        if (data[i][1] > posPredInterval[i][1] || data[i][1] < negPredInterval[i][1])
        {
            flag = true;
            break;
        }
    }
    return flag;
}
```

```
        private static int countErrors(double[][] data)
        {
            int cnt = 0;
            LinearRegression regr = new LinearRegression(new Data(data));
            double[][] clearData = new double[data.Length][];
            double[][] posPredInterval = regr.getPosPredInterval();
            double[][] negPredInterval = regr.getNegPredInterval();

            for (int i = 0; i < data.Length; i++)
            {
                if (data[i][1] > posPredInterval[i][1] || data[i][1] < negPredInterval[i][1])
                {
                    cnt++;
                }
            }

            return cnt;
        }

        public static double[][] deleteErrors(double[][] data)
        {
            LinearRegression regr = new LinearRegression(new Data(data));
            double[][] clearData = new double[data.Length - countErrors(data)][];
            double[][] posPredInterval = regr.getPosPredInterval();
            double[][] negPredInterval = regr.getNegPredInterval();

            int j = 0;
            for (int i = 0; i < data.Length; i++)
            {
                if (data[i][1] > posPredInterval[i][1] || data[i][1] < negPredInterval[i][1])
                    continue;
                clearData[j] = new double[2];
                clearData[j][0] = data[i][0];
                clearData[j][1] = data[i][1];
                j++;
            }

            return clearData;
        }

    }
}
```

Plot.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using System.Drawing;

namespace Regression
{
    class Plot
    {
        public Plot(RegressionModel regression, Chart chart)
        {
            int indX = 0;
            int indY = 1;
            double[][] graph = regression.getRegression();
            double[][] posConfInterval = regression.getPosConfInterval();
            double[][] negConfInterval = regression.getNegConfInterval();
            double[][] posPredInterval = regression.getPosPredInterval();
```

```
            double[][] negPredInterval = regression.getNegPredInterval();
            int N = graph.Length;

            //ensure that the chart is empty
            chart.Series.Clear();

            chart.Series.Add("Y").Color = Color.Red;
            chart.Series.Add("Y+ conf").Color = Color.Purple;
            chart.Series.Add("Y- conf").Color = Color.Purple;
            chart.Series.Add("Y+ pred").Color = Color.Blue;
            chart.Series.Add("Y- pred").Color = Color.Blue;

            for (int i = 0; i < 5; i++)
            {
                chart.Series[i].ChartType = SeriesChartType.Line;
            }

            chart.Legends.Clear();

            for (int i = 0; i < N; i++)
            {
                chart.Series[0].Points.AddXY(graph[i][indX], graph[i][indY]);
                chart.Series[1].Points.AddXY(posConfInterval[i][indX],
posConfInterval[i][indY]);
                chart.Series[2].Points.AddXY(negConfInterval[i][indX],
negConfInterval[i][indY]);
                chart.Series[3].Points.AddXY(posPredInterval[i][indX],
posPredInterval[i][indY]);
                chart.Series[4].Points.AddXY(negPredInterval[i][indX],
negPredInterval[i][indY]);
            }

            foreach (Series sr in chart.Series)
            {
                foreach (DataPoint point in sr.Points)
                {
                    point.MarkerStyle = MarkerStyle.Circle;
                    point.MarkerSize = 3;
                }
            }

            int round = 1000;
            chart.ChartAreas[0].AxisY.Maximum = Math.Round(posPredInterval[N - 1][1] + round);
            chart.ChartAreas[0].AxisY.Minimum = 0;
        }
    }
}

LinearRegression.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Regression
{
    class LinearRegression : RegressionModel
    {
        private double t = 2.02;

        public LinearRegression(Data _data) : base(_data)
        {
        }
```

```
        public override double getB0()
        {
            return (data.getSumY() - getB1() * data.getSumX()) / data.getN();
        }

        public override double getB1()
        {
            int N = data.getN();
            return (N * data.getSumXMultiplyY() - data.getSumX() * data.getSumY()) /
                (N * data.getSumXSquared() - data.getSumX() * data.getSumX());
        }

        public double getConfIntervalDelta(double x)
        {
            return t * getS() * (double)Math.Sqrt(1/ data.getN() + Math.Pow(x -
data.getAverageX(), 2) / (getSumXMinusAvgXSquared()));
        }

        private double getSumXMinusAvgXSquared()
        {
            double sum = 0;
            double avg = data.getAverageX();
            for (int i = 0; i < data.getN(); i++)
            {
                double arg = data.getX(i) - avg;
                sum += arg * arg;
            }
            return sum;
        }

        private double getS()
        {
            return (double)Math.Sqrt(getSumYMinusYCalcSquared() / (data.getN() - 2));
        }

        private double getSumYMinusYCalcSquared()
        {
            double sum = 0;
            for (int i = 0; i < data.getN(); i++)
            {
                double arg = data.getY(i) - getYCalculated(data.getX(i));
                sum += arg * arg;
            }
            return sum;
        }

        public double getPredIntervalDelta(double x)
        {
            return t * getS() * Math.Sqrt(1 + 1 / data.getN() + Math.Pow(x -
data.getAverageX(), 2) / (getSumXMinusAvgXSquared()));
        }

        public override double getYCalculated(double x)
        {
            return getB1() * x + getB0();
        }

        public override double[][] getRegression()
        {
            double[][] regression = new double[data.getN()][];
            for(int i=0; i < data.getN(); i++)
            {
                regression[i] = new double[2];
                //get current x
                regression[i][0] = data.getX(i);
```

```
            //calculate y using regression model
            regression[i][1] = getYCalculated(regression[i][0]);
        }

        return regression;
    }

    public override double[][] getPosConfInterval()
    {
        double[][] positiveConfInterval = new double[data.getN()][];
        for (int i = 0; i < data.getN(); i++)
        {
            positiveConfInterval[i] = new double[2];
            //get current x
            double x = data.getX(i);
            positiveConfInterval[i][0] = x;
            //calculate y using regression model
            positiveConfInterval[i][1] = getYCalculated(x) + getConfIntervalDelta(x);
        }

        return positiveConfInterval;
    }

    public override double[][] getNegConfInterval()
    {
        double[][] negativeConfInterval = new double[data.getN()][];
        for (int i = 0; i < data.getN(); i++)
        {
            negativeConfInterval[i] = new double[2];
            //get current x
            double x = data.getX(i);
            negativeConfInterval[i][0] = x;
            //calculate y using regression model
            negativeConfInterval[i][1] = getYCalculated(x) - getConfIntervalDelta(x);
        }

        return negativeConfInterval;
    }

    public override double[][] getPosPredInterval()
    {
        double[][] positivePredInterval = new double[data.getN()][];
        for (int i = 0; i < data.getN(); i++)
        {
            positivePredInterval[i] = new double[2];
            //get current x
            double x = data.getX(i);
            positivePredInterval[i][0] = x;
            //calculate y using regression model
            positivePredInterval[i][1] = getYCalculated(x) + getPredIntervalDelta(x);
        }

        return positivePredInterval;
    }

    public override double[][] getNegPredInterval()
    {
        double[][] negativePredInterval = new double[data.getN()][];
        for (int i = 0; i < data.getN(); i++)
        {
            negativePredInterval[i] = new double[2];
            //get current x
            double x = data.getX(i);
            negativePredInterval[i][0] = x;
            //calculate y using regression model
```

```
                negativePredInterval[i][1] = getYCalculated(x) - getPredIntervalDelta(x);
            }

            return negativePredInterval;
        }
    }
}

NonlinearRegression.cs
using System;

namespace Regression
{
    class NonlinearRegression : RegressionModel
    {
        private LinearRegression linearRegression;

        public NonlinearRegression(LinearRegression _linearRegression) : base(new
Data(Normalizer.reverseData(_linearRegression.getData().getData())))
        {
            linearRegression = _linearRegression;
        }

        public override double[][] getNegConfInterval()
        {
            return Normalizer.reverseData(linearRegression.getNegConfInterval());
        }

        public override double[][] getNegPredInterval()
        {
            return Normalizer.reverseData(linearRegression.getNegPredInterval());
        }

        public override double[][] getPosConfInterval()
        {
            return Normalizer.reverseData(linearRegression.getPosConfInterval());
        }

        public override double[][] getPosPredInterval()
        {
            return Normalizer.reverseData(linearRegression.getPosPredInterval());
        }

        public override double[][] getRegression()
        {
            double[][] regression = new double[data.getN()][];
            for (int i = 0; i < data.getN(); i++)
            {
                regression[i] = new double[2];
                //get current x
                regression[i][0] = data.getX(i);
                //calculate y using regression model
                regression[i][1] = getYCalculated(regression[i][0]);
            }

            return regression;
        }

        public override double getYCalculated(double x)
        {
            return Math.Pow(x, getB1()) * Math.Exp(getB0());
        }

        public override double getB0()
        {
```

```
            return linearRegression.getB0();
        }

        public override double getB1()
        {
            return linearRegression.getB1();
        }

        public double getPredIntervalDelta(double x)
        {
            double normalizedX = Normalizer.normalize(x);
            return Normalizer.reverse(linearRegression.getYCalculated(normalizedX) +
linearRegression.getPredIntervalDelta(normalizedX)) - getYCalculated(x);
        }

        public double getConfIntervalDelta(double x)
        {
            double normalizedX = Normalizer.normalize(x);
            return Normalizer.reverse(linearRegression.getYCalculated(normalizedX) +
linearRegression.getConfIntervalDelta(normalizedX)) - getYCalculated(x);
        }
    }
}
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Програмне забезпечення для оцінювання розміру мобільних застосунків, що створюються мовою Kotlin.

Позначення: «Regression».

1.2 Програмне забезпечення, необхідне для функціонування програми

Для функціонування програмного забезпечення «Regression» необхідною умовою є наявність .NET Framework версії 4.6 або вище.

1.3 Мови програмування

Програмне забезпечення для оцінювання розміру мобільних застосунків на Kotlin розроблене об’єктно-орієнтовною мовою програмування C#.

2 Функціональне призначення

Програмне забезпечення «Regression» було розроблене для автоматизації процесу оцінювання розміру мобільних застосунків та виконує наступні перелічені функції:

- внесення вхідних даних про проекти мобільних застосунків (кількість рядків коду та кількість класів);
- застосування нормалізуючого перетворення на основі натурального логарифму до внесених даних;

- побудова лінійної регресійної моделі на основі отриманих нормалізованих даних;
- побудова довірчого інтервалу та інтервалу передбачення для лінійної регресії;
- побудова нелінійної регресійної моделі з використанням зворотного нормалізуючого перетворення на основі лінійної регресійної моделі;
- розрахунок довірчого інтервалу та інтервалу передбачення для нелінійної регресії за допомогою зворотного нормалізуючого перетворення;
- побудова лінійного рівняння регресії для вхідних емпіричних даних у припущенні про нормальність їх розподілу;
- розрахунок метрик якості для лінійної та нелінійної регресійних моделей;
- отримання оцінки розміру мобільного застосунку;
- виведення отриманих розрахунків на екран;

3 Технічні засоби, що використовуються

Програмне забезпечення для оцінювання розміру мобільних застосунків на Kotlin призначене для запуску на персональних комп'ютерах з встановленою операційною системою Windows XP/7/8/8.1/10.

Робота з програмним забезпеченням передбачає наступні мінімальні апаратні вимоги:

- процесор: 32-розрядний процесор (x86), 1ГГц;
- обсяг оперативної пам‘яті: 2 Гб;
- обсяг вільного дискового простору: 500 Мб.
- наявність наступних пристроїв: монітор, клавіатура, миша;

5 Виклик та завантаження

Запуск програмного забезпечення відбувається за допомогою файлу запуску Regression.exe.

6 Вхідні дані

Вхідні дані для програми мають вводитись у вигляді файлу, який містить наступні стовбці значень: кількість рядків коду, кількість класів.

Також користувач має вводити вручну значення кількості класів для обчислення оцінки розміру мобільного застосунку.

7 Вихідні дані

Вихідними даними є результат обчислень у вигляді графіків та числових значень, що відображаються на екрані користувача.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА

Для запуску програмного забезпечення необхідно запустити файл «Regression.exe». Після запуску програмного забезпечення користувач бачить інтерфейс, що наведений на рис. Г.1.

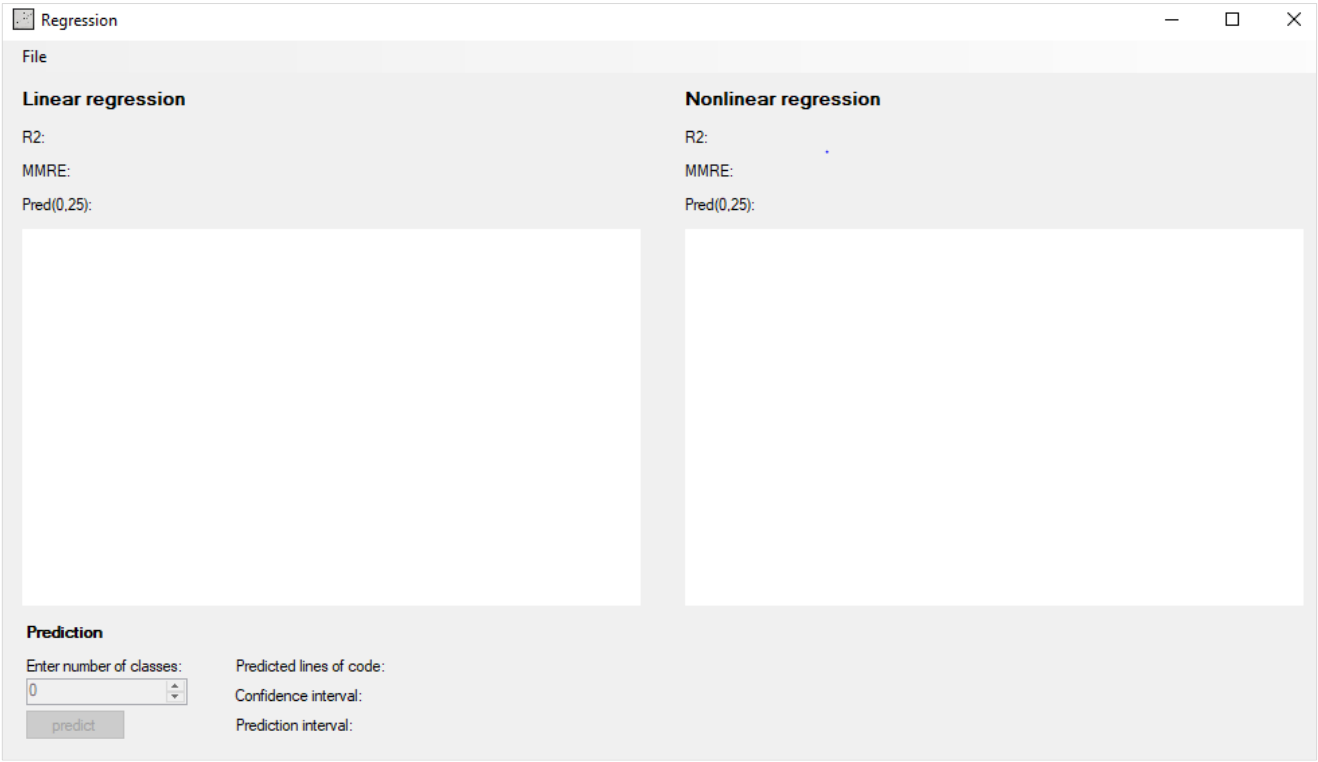

Рисунок Г.1 – Інтерфейс користувача програмного забезпечення

Користувач має можливість завантажити файл з вхідними даними. Також програмне забезпечення надає можливість здійснити нормалізацію вхідних даних, видалити викиди за наявності, побудувати лінійне рівняння регресії, побудувати довірчий інтервал та інтервал передбачення, побудувати нелінійну регресію та відповідні інтервали, розрахувати показники оцінки якості регресії, а також здійснити оцінювання розміру мобільних застосунків, але лише після завантаження файлу з вхідними даними.

Для того щоб ввести вхідні дані, користувач має натиснути на пункт меню «File», потім обрати пункт «Load», після чого необхідно обрати файл з розширенням .csv.

У випадку якщо користувач завантажить файл з некоректним розширенням, або структура даних у файлі буде неправильною, то користувач отримає повідомлення про помилку, яке зображено на рис. Г.2.

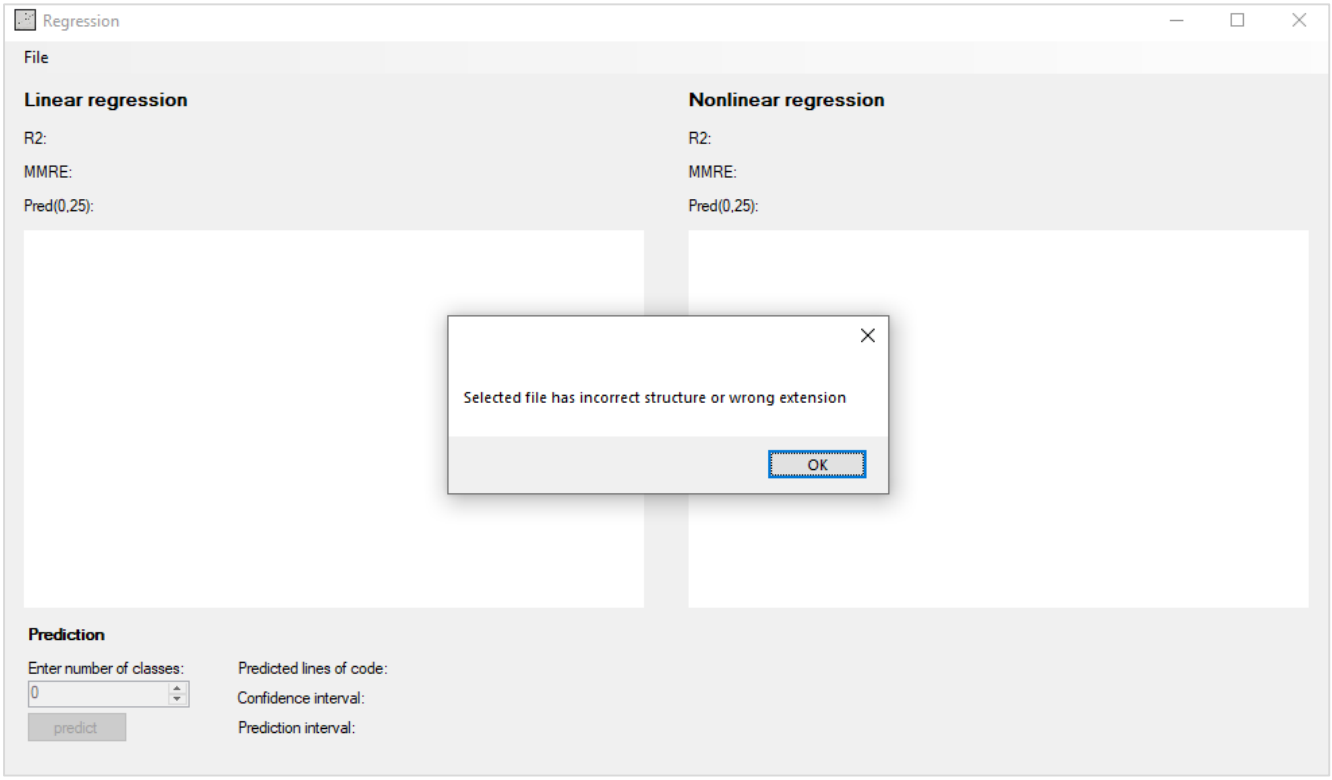

Рисунок Г.2 – Повідомлення про помилку

Якщо вхідні дані завантажено успішно, програмне забезпечення застосує до вхідних даних нормалізуюче перетворення на основі натурального логарифму та видалить викиди за наявності. Також буде здійснена побудова лінійної регресійної моделі у припущенні про нормальність розподілу вхідних даних, на графіку буде відображена лінійна регресійна модель, довірчі інтервали та інтервали прогнозування. Після цього на основі побудованого лінійного рівняння регресії буде побудована нелінійна регресійна модель та відповідні інтервали, які також будуть відображені на графіку.

Для побудованих регресійних моделей буде здійснено розрахунок показників якості R^2, MMRE, Pred(0,25). Також після побудови нелінійної регресійної моделі функція оцінки розміру мобільного застосунку стане активною та доступною для використання. Виведені на екран результати розрахунків після завантаження вхідних даних зображені на рис. Г.3.

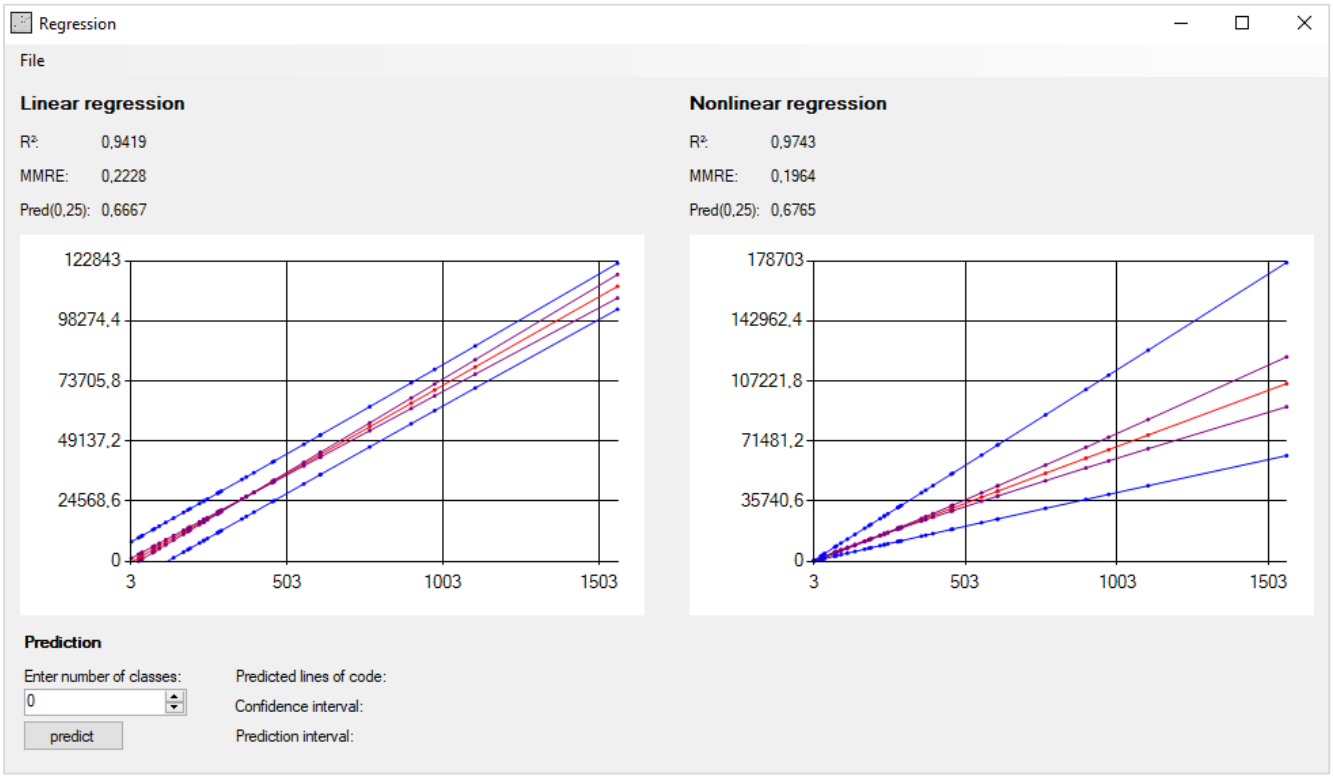

Рисунок Г.3 – Результати розрахунків

Для розрахунку оцінки розміру мобільного застосунку потрібно ввести відповідну кількість класів у поле, що знаходиться під надписом «Enter number of classes», та натиснути кнопку «predict». Результат оцінювання розміру мобільного застосунку представлено на рис. Г.4.

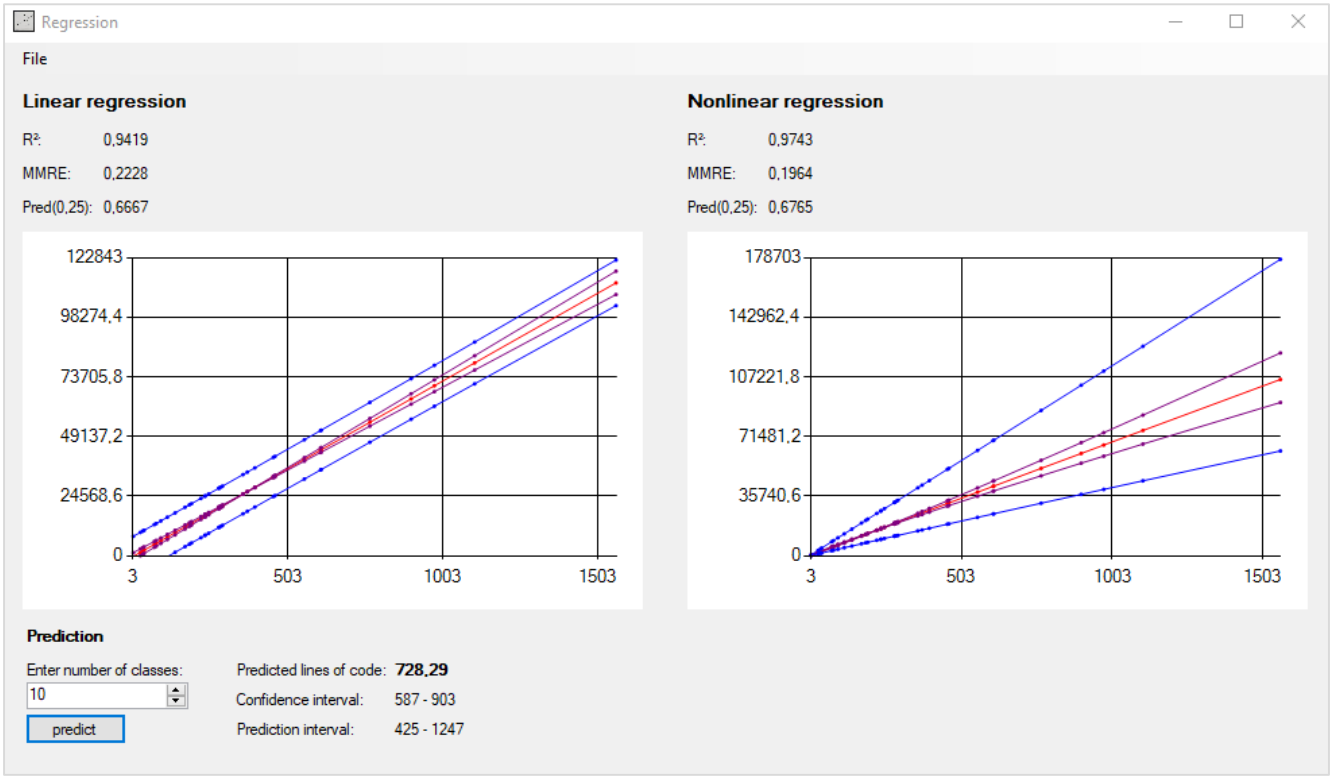

Рисунок Г.4 – Результат оцінювання розміру мобільного застосунку

У результаті розрахунку оцінки розміру мобільного застосунку на екран буде виведена розрахована за нелінійним рівнянням регресії оцінка кількості рядків коду, а також довірчий інтервал та інтервал прогнозування для отриманої оцінки.

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення для оцінювання розміру мобільних застосунків що створюються мовою Kotlin, яке було розроблено в рамках даної кваліфікаційної роботи.

2 Мета випробувань

Метою проведення випробування є перевірка працездатності програмного забезпечення, а також перевірка відповідності вимогам, сформульованим у технічному завданні.

3 Вимоги до програми

При проведенні випробувань необхідно перевірити чи відповідають функціональні характеристики розробленого програмного забезпечення вимогам, що викладені у пункті «Вимоги до складу виконуваних функцій» Технічного завдання.

4 Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

- технічне завдання;
- текст програми;
- опис програми;
- інструкція користувача;
- програма та методика випробувань.

5 Засоби та порядок випробувань

Вимоги до складу та параметрів технічних і програмних засобів вказані у Технічному завданні (Додаток А). Випробування проводилось на платформі з операційною системою Windows 10 64bit.

Порядок проведення випробувань передбачає виконання функцій та проведення аналізу отриманих результатів. Необхідно перевірити відповідність функціональних характеристик програмного забезпечення вимогам, що викладені у програмній документації.

6 Методика випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог, визначених у програмній документації, здійснюється візуально. У ході проведення випробувань необхідно перевірити відповідність складу програмної документації, що надає розробник, з переліком програмної документації, наведеним у пункті «Вимоги до програмної документації» цього документу.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Проведення перевірки працездатності програмного забезпечення виконується згідно з пунктом «Вимоги до функціональних характеристик», наведеним у технічному завданні.

Якщо склад та послідовність дій, що виконуються при перевірці, відповідають функціональним вимогам, викладеним у технічному завданні, то перевірка вважається успішно завершеною.

В процесі тестування була здійснена перевірка функціональності програмного забезпечення для оцінювання розміру мобільних застосунків що створюються мовою Kotlin. У табл. Д.1 наведений перелік випробувань основних функціональних можливостей.

Таблиця Д.1 – Методика випробувань

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	Завантаження файлу із розширенням .xlsx	Повідомлення про помилку, функція передбачення розміру мобільних застосунків є неактивною	Виконано успішно	
2	Завантаження файлу із розширенням .csv з некоректною структурою даних	Повідомлення про помилку, функція передбачення розміру мобільних застосунків є неактивною	Виконано успішно	
3	Завантаження файлу із розширенням .csv з коректною структурою даних	Дані успішно введено, функція передбачення розміру мобільних застосунків стала доступною для виконання	Виконано успішно	

Для кожної з наведених дії потрібно провести випробування та перевірити правильність виконання вказаних функцій.