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
_____
«___» _____ 2022 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «магістр»

на тему: **«**Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, та розробка програми для її реалізації.**»**

Виконала: студентка групи 6151м
_____Телехан А. М.
(підпис, ПІБ)

Керівник роботи:
доцент кафедри ПЗАС, к.т.н.,
доцент
(посада, науковий ступень, вчене звання)
_____ Макарова Л.М.
(підпис, ПІБ)

Миколаїв – 2022 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
Освітній ступень Магістр
Галузь 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____проф. С.Б.Приходько
(підпис)
«10» жовтня 2022 р.

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

Студенту Телехан Анні Миколаївні
(Прізвище, ім'я, по батькові)

1. Тема роботи: «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, та розробка програми для її реалізації»

Керівник роботи Макарова Лідія Миколаївна

Затверджені наказом ректора № 798 уч від «28» 09 2022 р.

2. Термін подання роботи: 14.11.2022 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

_____

_____

_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 10.10.2022 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	11.10.2022	НС*
2.	Підготовка розділу з огляду літератури та обґрунтування необхідності проведення досліджень за обраною темою	12.10.2022	НС*
3.	Підготовка розділу (ів) з результатів власних досліджень	13.10.2022	НС*
4.	Підготовка розділу з проекту програмного забезпечення	03.11.2022	
5.	Підготовка організаційно-економічного розділу	06.11.2022	
6.	Підготовка розділу з охорони праці	09.11.2022	
7.	Підготовка розділу з охорони навколишнього середовища	11.11.2022	
8.	Підготовка розділу Висновки	12.11.2022	
9.	Оформлення списку використаних джерел та додатків	13.11.2022	
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.)	14.11.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
11.	Підготовка презентації та доповіді	19.11.2022	
12.	Попередній захист роботи на засіданні кафедри ПЗАС	20.11.2022	
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	28.11.2022	

* - за результатами наукового стажування (НС), яке було з 01.09.2022 по 09.10.2022 р.

Студент _____ Телехан А. М.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л. М.
(підпис) (ПІБ)

РЕФЕРАТ

Телехан Анна Миколаївна

Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, та розробка програми для її реалізації

Кваліфікаційна робота на здобуття освітнього рівня магістра зі спеціальності 121 – «Інженерія програмного забезпечення». Національний університет кораблебудування імені адмірала Макарова. Миколаїв, 2022 р.

Обсяг роботи: 128 стор., 22 табл., 12 рисунки, 26 використаних джерел, 5 додатків.

Актуальність теми. Процес аналізу та оцінки розміру програмного забезпечення є основою для подальшого розрахунку ресурсів та трудомісткості, необхідних для розробки програмного забезпечення та реалізації проекту. Проте моделей для оцінювання розміру персональних застосунків, що створюються мовою Kotlin для Андроїд майже не існує. Таким чином, побудова нелінійної регресійної моделі, яка дасть змогу швидко та вірогідно оцінювати розмір персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд є актуальним завданням

Мета дослідження. Метою даної роботи є підвищення достовірності оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Об’єкт дослідження: процес оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Предмет дослідження: однофакторна нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Наукова новизна одержаних результатів: удосконалено однофакторну нелінійну регресійну модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, на основі нормалізації за допомогою десяткового логарифма та викидів регресії. Ця нелінійна регресійна модель у порівнянні з існуючою моделлю має більшу достовірність оцінювання по параметрам якості *MMRE* і *PRED*(0,25) та вужчі довірчий інтервал і інтервал передбачення.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 р.).

Публікації. Результати кваліфікаційної роботи висвітлено в 1 науковій праці – тезах конференції.

Ключові слова: оцінювання розміру, персональний застосунок-органайзер, нелінійна регресійна модель, андроїд, kotlin, нормалізуюче перетворення.

ABSTRACT

Telekhan Anna Mykolaivna

Nonlinear regression model for estimating the size of personal applications created for Android using the Kotlin programming language and developing software for its implementation.

Qualification work for obtaining a master's degree in Specialty 121 - Software Engineering. Admiral Makarov National University of Shipbuilding. Mykolaiv, 2022.

The work contains: 128 pages, 22 tables, 12 figures, 26 references, 5 appendices.

Relevance of the topic. An important step for further estimating resources required for software development and project planning is the process of analyzing and estimating the software size. However, there are almost none of universal methods or models, which are optimal for estimating the size of personal applications, created in Kotlin. Therefore, the task of building the nonlinear regression model for estimating the size of personal organizers, created using the Kotlin programming language for Android, is relevant.

The study's purpose: to increase the reliability of estimating the size of personal applications created for Android, using the Kotlin programming language.

The object of the study: the process of estimating the size of personal applications created for Android, using Kotlin programming language.

The subject of the study: a univariate nonlinear regression model for estimating the size of personal applications created for Android, using Kotlin programming language.

The scientific novelty of the obtained results: an one-factor nonlinear regression model has been improved to estimate the size of personal organizer applications created in the Kotlin language for Android, based on normalization using the decimal logarithm and regression emissions. This nonlinear regression model, compared to the existing model, has greater reliability of estimation by *MMRE* and *PRED*(0,25) quality parameters and a narrower confidence interval and prediction interval.

The practical value of the results. As part of the qualification work, a program for estimating the size of personal organizer applications was developed.

Approbation of research results. The main research results were tested at the III All-Ukrainian scientific and practical Internet conference "Information Technologies: Models, Algorithms, Systems" (Mykolaiv, October 26-28, 2022).

Publications: The main results of the qualification work are presented in 1 scientific work – theses of the conference.

Keywords: ESTIMATING THE SOFTWARE SIZE, PERSONAL ORGANIZERS, NONLINEAR REGRESSION MODEL, ANDROID, KOTLIN, NORMALIZING TRANSFORMATION

ЗМІСТ

ВСТУП

Актуальність теми. При розробці будь-якого програмного забезпечення пред’являється низка вимог і вирішуються задачі оцінювання, і персональні Андроїд застосунки не є виключенням. Процес оцінювання є першим кроком та основою для подальшого планування програмних проектів. Перш за все для успішної реалізації програмного забезпечення (ПЗ) необхідні задачі оцінювання трудомісткості та ресурсів, оскільки максимально достовірне оцінювання необхідних трудових витрат на написання коду ПЗ є однією з основних проблем в сфері управління проектами розробки ПЗ [1]. При розробці програмних проектів одного класу найбільший вплив на трудомісткість має розмір застосунків так, як оцінювання розміру програмних продуктів є основою для подальшого визначення тривалості розробки, трудомісткості, а також вартості програмного проекту [2].

Існують різні методи оцінювання розміру програмних продуктів. Деякі з них є універсальними, інші створені для планування певних класів проектів. Проте в інженерії програмного забезпечення не існує єдиних рекомендацій для оцінювання розміру програмних продуктів. Серед сучасних методів та моделей, котрі використовують для отримання оцінки розміру ПЗ, можна виділити [3]:

– метод конструктивної вартості;

– метод функціонально-бальної оцінки;

– експертне оцінювання;

– нейронні мережі;

– нечітку логіку;

– регресійний аналіз;

– комбіновані методи та інші.

Але аналіз сучасних методів та моделей оцінювання розміру ПЗ показав, що на сьогодні, більшість з математичних моделей, що дозволяють оцінити

розмір програмних продуктів, стають непридатними до використання через інший клас задач, відсутність даних, ресурсів або експертних навичок в цій галузі. Сучасні методи часто не враховують нелінійність вихідних даних або не забезпечують необхідну достовірність [3].

Таким чином, побудова нелінійної регресійної моделі, яка дасть змогу швидко та вірогідно оцінювати розмір персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд є актуальним завданням. Побудована регресійна модель дозволить підвищити точність та швидкість оцінювання розміру персональних застосунків, що створюються мовою Kotlin для Андроїд.

Метою даної роботи є підвищення достовірності оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

– провести аналіз сучасних методів та моделей для оцінювання розміру персональних застосунків-органайзерів на Андроїд;

– обґрунтовати необхідність проведення дослідження;

– провести аналіз на викиди та отримати лінійне рівняння регресії;

– побудовати лінійну регресійну модель, довірчі інтервали та інтервали передбачення для негаусівського набору даних (кількості рядків коду та кількості класів);

– на основі нормалізованих даних з використанням нормалізуючого перетворення на основі десяткового логарифму побудувати нелінійну регресійну модель;

– розробити програму для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Об’єктом дослідження є процес оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Предметом дослідження є однофакторна нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Методи дослідження: для розв'язання поставлених задач були застосовані методи теорії ймовірностей, математичної статистики математичного моделювання, регресійного аналізу, об`єктно-орієнтованого програмування.

Наукова новизна одержаних результатів: удосконалено однофакторну нелінійну регресійну модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, на основі нормалізації за допомогою десяткового логарифма та викидів регресії. Ця нелінійна регресійна модель у порівнянні з існуючою моделлю має більшу достовірність оцінювання по параметрам якості *MMRE* і *PRED*(0,25) та вужчі довірчий інтервал і інтервал передбачення.

Практичне значення одержаних результатів. В межах кваліфікаційної роботи було розроблено програму для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Особистий внесок здобувача. Кваліфікаційна робота є самостійно виконаною працею. Усі результати, викладені у роботі, отримані автором особисто. У праці, яка опублікована у співавторстві з керівником роботи [4], здобувачеві належить: побудова однофакторної нелінійної регресійної моделі для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

Апробація результатів досліджень. Основні положення і результати досліджень, викладені у кваліфікаційній роботі, пройшли апробацію на III Всеукраїнській науково-практичній інтернет-конференції «Інформаційні технології: моделі, алгоритми, системи» (м. Миколаїв, 26-28 жовтня 2022 р.).

Публікації. Результати роботи висвітлено у 1 науковій праці – тезах конференції.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ І МОДЕЛЕЙ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПЕРСОНАЛЬНИХ ЗАСТОСУНКІВ-ОРГАНАЙЗЕРІВ ДЛЯ АНДРОЇД

1.1 Мова програмування Kotlin для створення застосунків на Андроїд

Kotlin – це об'єктно-орієнтована, кросплатформна, статично типізована, універсальна мова програмування компанії JetBrains (2016). Мова програмування Kotlin працює поверх віртуальної машини Java (Java Virtual Machine) та компілюється в байт-код. Тобто додаток на Kotlin є повністю сумісним з Java, що поступово дозволяє багатьом розробникам переходити до використання Kotlin при розробці програмних продуктів. В основному мова Kotlin націлена на використання віртуальної машини Java, але також цілком можливе компілювання в JavaScript і запуск в браузерах.

Мову програмування Kotlin використовують для створення персональних та веб-застосунків, що можуть працювати на Windows, Linux, Mac OS, iOS, Андроїд [5]. У сфері розробки програмного забезпечення для Андроїд вона поступово замінює мову Java. Мова Kotlin є офіційно підтримуваною (компанією Google) та пріоритетною мовою для розробки персональних застосунків на Андроїд [6].

Синтаксис Kotlin при створенні зазнав впливу таких мов, як JavaScript, Java, Scala, Groovy, C# та Python. Творці Kotlin прагли створити лаконічнішу, та простішу типово-безпечну мову в порівнянні з Java та Scala. Саме тому, в наслідок бажаного спрощення отримали мову програмування, котра має швидшу компіляцію та кращу підтримку середовища розробки. Kotlin підтримує об'єктно-орієнтоване та процедурне програмування з використанням функцій, що робить мову більш універсальною. Вона має ясний і зрозумілий синтаксис і є досить легкою для навчання [5]. Повна сумісність з Java також

дозволяє Kotlin впроваджувати нові функції в існуючі Андроїд та iOS застосунки без повного переписування програмного продукту. Мова Kotlin є Open Source проектом, вихідний код даної мови програмування знаходиться у відкритому доступі, котрий можна знайти в репозиторії на Github [7].

Андроїд – це повноцінна операційна система і платформа, створена на базі ядра Linux V3.6 компанією Google. Перші версії Андроїд знайшли своє застосування в сегменті смартфонів і планшетів. Однак повний спектр обчислювальних сервісів і багаті функціональні можливості Андроїд дозволяють створювати застосунки, які виходять далеко за рамки сегмента персональних телефонів. Застосунки під ОС Андроїд є програмами в нестандартному байт-коді для віртуальної машини Dalvik, для яких було розроблено формат інсталяційних пакетів .APK. Андроїд, як і Kotlin, є Open Source проектом. Вихідний код Андроїд доступний на сайті проекту Андроїд Open Source Project (AOSP) [8], який підтримується Google.

1.2 Метрики програмного забезпечення

Метрика програмного забезпечення – це міра, що дозволяє отримати чисельне значення певної властивості програмного забезпечення або його специфікацій [9]. Часто методи аналізу метрик ПЗ порівнюють з методами та моделями для оцінювання розміру, витрат і зусиль. Між кількісними та якісними метриками на практиці набагато частіше використовують саме кількісні метрики, оскільки вони легше піддаються підрахунку та статистичній обробці. Тож даний підхід зазвичай намагаються переносити і в розробку програмних продуктів.

Серед найпоширеніших методів аналізу метрик програмного забезпечення можна виділити методи за [10]:

– кількістю рядків коду;

– кількістю класів;

– кількістю модулів;

– цикломатичною складністю;

– аналізу функціональних точок;

– кількістю помилок на 1000 рядків коду;

– ступенем покриття коду тестуванням;

– покриттям вимог;

– метрики Холстеда та інші.

Зазвичай перераховані вище методи аналізу метрик використовуються як основа для розрахунку інших показників програмних проектів таких, як трудомісткість, тривалість розробки чи вартість програмного проекту. На етапі планування будь-яких проектів важливим етапом є визначення розміру програмного забезпечення, що розроблюється. Найбільш поширеною метрикою вихідного коду для відображення розміру програмного продукту є кількість рядків коду SLOC (Source Lines Of Code).

Кількість рядків коду (Source Lines Of Code) – це розмірно-орієнтована метрика програмного забезпечення, для якої об'єм ПЗ розраховується виходячи з кількості рядків в тексті вихідного коду [10].

Спочатку SLOC застосовувався в умовах, коли мови програмування були відносно прості за структурою і для них характерним було відповідність одного рядка коду одній команді мови. Серед методик урахування кількості рядків коду є дві основні [9]:

– Кількість фізичних SLOC (використовувані абревіатури: LOC, SLOC, KLOC, KSLOC, DSLOC) – визначається як загальна кількість рядків коду, з урахуванням коментарів і порожніх рядків (при вимірюванні показника на кількість порожніх рядків, як правило, вводиться обмеження – враховується кількість, що не перевищує 25% загальної кількості рядків у вимірюваному блоці коду). Літера "D" (Delivered) в абревіатурі означає «поставлений» код, тобто тільки той, що увійшов до закінченого програмного продукту. Дуже часто виникає необхідність враховувати не тільки поставлений код, але і

допоміжний або проміжний, який не входить до закінченого продукту, але що потрібен для реалізації проекту і вимагає певних витрат праці.

– Кількість логічних SLOC (використовувані абревіатури: LSI, DSI, KDSI, де SI – Source Instructions) – визначається як кількість команд і залежить від використовуваної мови програмування. У випадку, коли мова програмування не підтримує розміщення кількох команд в одному рядку, кількість логічних SLOC буде відповідати числу «фізичних» рядків коду, за винятком порожніх рядків та рядків коментарів. Якщо ж мова програмування дозволяє розміщення кількох команд на одному рядку, то один «фізичний» рядок повинний бути врахований як декілька «логічних», якщо він містить більше однієї команди мови, що використовується.

Також існує багато похідних від кількості рядків коду метрик, котрі дозволяють оцінити окремі показники проекту [9]:

– кількість пустих рядків (Blank Lines Of Code, BLOC);

– кількість рядків, що містять коментарі (Comment Lines Of Code, CLOC);

– кількість рядків, що містять вихідний код і коментарі (Lines with Both Code and Comments, C&SLOC);

– кількість рядків, що містять декларативний вихідний код;

– кількість рядків, що містять імперативний вихідний код;

– відсоток коментарів (кількість рядків коментарів, помножене на 100 і поділена на кількість рядків коду);

– середня кількість рядків для функцій (методів);

– середня кількість рядків, що містять вихідний код для функцій (методів);

– середня кількість рядків для модулів;

– середня кількість рядків для класів.

При цьому, оцінювання в кількості рядків коду має ряд переваг. Наприклад, метрики за кількістю рядків в завершених проектах чи модулях

програм можуть бути легко зібрані за допомогою службових засобів інтегрованих середовищ розробки (IDE) чи спеціальних програм [10].

1.3 Аналіз сучасних методів та моделей для оцінювання розміру програмного забезпечення

Існує велика кількість різних методів та моделей для оцінювання розміру програмних застосунків. Деякі з них створені з думкою про певний вид програмного продукту, інші ж створюються більш універсальними. Згідно однієї з багатьох класифікацій методи та моделі оцінювання розміру програмного забезпечення відносяться до наступних класів [3]:

– методи та моделі експертної оцінки;

– методи оцінки по математичним моделям;

– комбіновані методи - поєднують два або більше методів для отримання більш точної оцінки.

Методи та моделі експертної оцінки застосовуються при відсутності дискретних емпіричних даних. Фактично єдиним обґрунтуванням оцінки є думка експерта чи групи експертів, які мають досвід у розробці певного виду програмного забезпечення [3].

До методів оцінки по математичним моделям відносяться [11]:

– Нейронні мережі. Використання нейронних мереж дозволяє оцінити розмір програмного проекту за допомогою використання штучного інтелекту та подальшого навчання нейронної мережі.

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

– Генетичні алгоритми. Покращений метод на базі методу нейронних мереж.

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

– Регресійні моделі. Розмір програмного проекту обчислюється на основі математичної моделі та заданих вхідних параметрів.

Розглянемо детальніше метод регресійних моделей для оцінювання розміру програмного забезпечення. Існують лінійні та нелінійні моделі оцінювання розміру. В лінійних регресійних моделях знаходять коефіцієнти рівняння регресії, використовуючи емпіричні дані, та за допомогою отриманого рівняння виконують прогнозування значень розміру.

Якщо випадкові величини, що входять до моделі, не підпорядковуються нормальному закону розподілу, то маємо нелінійну регресійну модель. Під час побудови нелінійних регресійних моделей використовуються наступні методи: метод простого перебору, метод лінеаризуючих перетворень, метод нормалізуючих перетворень. Методом простого перебору знаходять нелінійну функцію регресії та на її основі будують нелінійну регресійну модель. Метод нормалізуючих перетворень полягає у застосуванні нормалізуючих перетворень випадкових змінних регресії, які дозволять отримати близький до лінійного зв’язок між нормалізованими змінними. Тобто 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 [13]. Даний метод позбавлений недоліків, які мають методи лінеаризуючих перетворень та простого перебору.

1.4 Обґрунтування необхідності проведення дослідження з оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд

Сучасні методи та моделі продовжують розвиватись весь час, та незважаючи на інтенсивні дослідження, залишається ряд невирішених аспектів і проблем. Через швидкий розвиток інформаційних технологій постійно маємо нові та підвищені вимоги до точності та оперативності оцінювання програмних проектів на ранніх стадіях розробки. Але поки не існує універсального способу для оцінки розміру персональних застосунків.

Аналіз сучасних моделей та методів показав, що на сьогодні, більшість з математичних моделей, що дозволяють оцінити розмір персональних застосунків, стають непридатними до використання через інший клас задач, відсутність даних, ресурсів або експертних навичок у сфері розробки. Сучасні методи часто не враховують нелінійність вихідних даних або не забезпечують необхідну достовірність. Результат оцінювання розміру ПЗ на основі регресійної моделі значною мірою залежить від мови програмування, обраної для розробки конкретного програмного продукту, та типу програмного проекту. Також, в реальних проектах дуже рідко чи майже неможливо вдається отримати гаусівські дані, і зв’язки між отриманими даними не завжди є лінійними.

Отже, існуючі засоби та методи не дозволяють з необхідною достовірністю оцінювати розмір програмного забезпечення з урахуванням особливостей розробки персональних застосунків, створених мовою програмування Kotlin для Андроїд. Тому побудова нелінійної регресійної моделі, яка дасть змогу швидко та достовірно оцінювати розмір персональних застосунків, що створюються мовою Kotlin для Андроїд, є актуальною задачею. Побудована модель дозволить підвищити достовірність прогнозування розміру персональних застосунків.

2 ПОБУДОВА НЕЛІНІЙНОЇ РЕГРЕСІЙНОЇ МОДЕЛІ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПЕРСОНАЛЬНИХ ЗАСТОСУНКІВ-ОРГАНАЙЗЕРІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN ДЛЯ АНДРОЇД

2.1 Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, із застосуванням нормалізуючих перетворень

Первинну обробку емпіричних даних виконують у наступному порядку:

1) Знаходять $\bar{x}$ (точкову оцінку математичного сподівання) n значень x_i ($i = 1, 2, \dots, n$).

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

2) Визначають вибіркову дисперсію S_x^2 (незміщену точкову оцінку дисперсії) та точкову оцінку середнього квадратичного відхилення $\hat{\sigma}_x$.

$$S_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - x)^2 \,;\; \hat{\sigma}_x = \sqrt{S_x^2} \tag{2.2}$$

3) Перевіряють нормальність закону розподілу емпіричних даних.

4) Визначають наявність викидів та, якщо останні знайдені, відповідні результати відкидають і повторюють обчислення.

Зазвичай викид визначають як спостереження, яке настільки відхиляється від інших спостережень, що викликає підозру в тому, що воно було породжене іншим механізмом. Точки даних, які не описуються регресійною моделлю, розглядаються як викиди.

Для визначення викидів у негаусівських даних на першому етапі рекомендується використовувати відповідний метод на основі квадрату відстані Махаланобіса (Mahalanobis) для нормалізованих даних [14, 15]. Для цього негаусівські дані нормалізують (бажано із застосуванням багатовимірних перетворень) та для них визначають значення квадрату відстані Махаланобіса, яке для кожної точки багатовимірних даних $i = 1,2, \dots, N$ позначається як d_i^2 і обчислюється за наступною формулою:

$$D^2 = (Z - \overline{Z})^T S_N^{-1} (Z - \overline{Z}), \tag{2.3}$$

або формула для i - го елемента:

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

де Z_i – i-точка багатовимірного даних гаусівського вектору Z;

$\bar{Z}$ – вектор вибіркових середніх багатовимірних даних гаусівського вектору Z (середні по нормалізованих факторах $X_1, X_2, \dots, X_n$);

S_N – матриця багатовимірних коваріацій.

Відповідно коваріаційна матриця розраховується за наступною формулою:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (Z - \overline{Z}) \, (Z - \overline{Z})^T. \tag{2.4}$$

Після обчислення d_i^2 можна застосовувати або критерій Пірсона $\chi 2$ або критерій Фішера F. Для використання критерію Фішера потрібно додатково для кожного d_i^2 розрахувати статистику:

$$T_{S_i} = (N - k)Nd_i^2/((N^2 - 1)k), \quad (2.5)$$

де N - всього елементів у вибірці;

k - кількість параметрів.

Якщо статистика T_{S_i} більше критичного значення Фішера $F_{k,N-k,\alpha}$, відповідного обраному рівню значущості α (у нас 0,05), то такі точки вважаються викидами. Їх потрібно прибрати з вибірки і заново повторити розрахунок.

Якщо дані слідують багатовимірному нормальному розподілу, то розподіл квадрата відстані Махаланобіса поводиться як розподіл χ^2. Для великих $N - m$ (принаймні 25) відстань повинна поводитися приблизно як незалежні $\chi^2_{m,\text{a}}$ – випадкові величини. Тут $\chi^2_{m,a}$ – це квантиль розподілу χ^2, α – рівень значущості.

Значення даних, для яких значення квадрата відстані Махаланобіса більше, ніж квантиль розподілу χ^2, вважаються викидами, і ці значення відкидаються. Після відкидання зменшена кількість точок багатовимірних даних нормалізується за допомогою нормалізуючого перетворення знову, поки всі значення квадрата відстані Махаланобіса не будуть меншими або рівними квантилю розподілу χ^2.

Багато статистичних критеріїв, методів і оцінок розроблені тільки для випадку нормального початкового розподілу. Тому при первинній обробці експериментальних даних перевіряють нормальність закону розподілу результатів спостережень.

У разі великої вибірки значень випадкової величини (коли $n > 30$) перевірку нормальності закону розподілу результатів

спостережень в тому числі виконують за критерієм Пірсона (критерієм χ^2) [16]. За цім критерієм спочатку обчислюють значення χ^2:

$$\chi^2 = \sum_{j=1}^{m} \frac{(x_j - np_j)^2}{np_j} \tag{2.6}$$

x_{min}, x_{max} – відповідно мінімальне і максимальне значення випадкової величини *X*;

n_j–абсолютна частота в *j*-му підінтервалі (кількість значень випадкової величини, які потрапляють у *j*-ий підінтервал);

p_j – ймовірність того, що значення випадкової величини *X* потрапляють у *j*-ий підінтервал.

Кількість підінтервалів (кліток), на які розбивається інтервал $[x_{min}, x_{max}]$ може бути визначений за наступними формулами: $m = log2\, n + 1 = 3{,}3lgn + 1$ (формула Старджеса) або $m = 5lgn$ (формула Брукса і Карузера).

Ймовірність того, що значення випадкової величини *X* потрапляють у *j*-ий підінтервал можуть бути визначені як:

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

де x_{j-1} та x_j – ліва і права границі *j*-го підінтервалу.

Після цього в залежності від рівня значимості α та кількості ступенів вільності ν за таблицею верхніх 100α відсоткових точок розподілу χ^2 знаходять значення $\chi^2_{кр}$. Якщо $\chi^2 \leq \chi^2_{кр}$, то з ймовірністю $1-\alpha$ гіпотеза про нормальний закон розподілу результатів спостережень прийнята, якщо $\chi^2 > \chi^2_{кр}$ – гіпотезу відкинута.

Кількість ступенів вільності ν визначається як $\nu = m - k - 1$, де k – кількість параметрів, від яких залежить закон розподілу. Для нормального розподілу $k = 2$.

За результатами збору метрик було розглянуто 49 проектів програмного забезпечення. Метрики було взято з проектів з відкритим вихідним кодом на репозиторії Github [6]. Було зібрано такі характеристики програмного забезпечення як кількість класів (Number Of Classes) – в якості незалежної змінної *X*, та кількість рядків коду в тисячах (Kilo Lines Of Code) – в якості залежної змінної *Y*. Результат збору метрик наведено у таблиці 2.1:

Таблиця 2.1 – Метрики програмного забезпечення

Номер проекту	NC (X)	KLOC (Y)	Номер проекту	NC (X)	KLOC (Y)
1	6	0,601	26	155	4,845
2	14	0,710	27	159	12,443
3	14	0,248	28	177	10,236
4	15	1,886	29	232	17,063
5	20	1,275	30	245	16,982
6	22	1,387	31	248	10,665
7	30	2,574	32	280	21,336
8	33	0,539	33	294	17,911
9	43	1,836	34	304	11,894
10	44	5,654	35	334	15,529
11	50	3,082	36	366	16,810
12	51	2,819	37	416	28,892
13	72	6,503	38	416	45,038
14	79	4,730	39	457	27,699
15	83	7,778	40	477	21,516
16	84	4,760	41	593	28,254
17	89	2,934	42	613	48,386
18	98	8,686	43	622	50,913
19	106	4,924	44	767	61,863
20	122	7,583	45	914	30,312
21	122	4,234	46	1026	111,063
22	124	5,395	47	1186	83,726
23	132	6,351	48	1241	87,704
24	139	11,747	49	2160	132,195
25	142	6,657			

Визначимо основні статистичні характеристики отриманої вибірки - оцінку математичного сподівання, дисперсії та середньоквадратичного відхилення за формулами (2.1), (2.2) по незалежній (*X*) та залежній змінній (*Y*): $\bar{x} = 314{,}612$, $S_x^2 = 167973{,}242$, $\hat{\sigma}_x = 409{,}845$, $\bar{y} = 20{,}778$, $S_y^2 = 854{,}073$, $\hat{\sigma}_y = 29{,}224$.

Перевіримо нормальність закону розподілу емпіричних даних. Для цього побудуємо гістограму розподілу емпіричних даних.

Розмах вибірки: $R = x_{max} - x_{min} = 2154$, $R = y_{max} - y_{min} = 131{,}947$.

За формулою Старджеса визначимо кількість інтервалів для побудови гістограми емпіричного розподілу: $m = log_2 N + 1 = 3{,}3\, lg\, N + 1 = 6{,}594 = 6$.

Оскільки графік щільності ймовірності нормального розподілу є унімодальним, при побудові гістограми слід обрати непарну кількість інтервалів. За формулою Старджеса отримали парне число, візьмемо значення на одиницю менше, тому $m = 5$.

Довжина інтервалу гістограми емпіричного розподілу: $h_x = \frac{R_x}{m} = \frac{2154}{5} = 430{,}8$, $h_y = \frac{R_y}{m} = \frac{131{,}947}{5} = 26{,}3894$.

Побудуємо гістограму емпіричного розподілу та функцію теоретичного розподілу на одному графіку. Для цього проінтегруємо функцію щільності ймовірності нормального розподілу на кожному з інтервалів за допомогою методу трапецій, розбивши кожен інтервал на десять частин. Також на основі визначених теоретичних ймовірностей на кожному з інтервалів гістограми виконаємо розрахунки за формулою (2.6) для кожного з інтервалів для визначення значення критерія Пірсона, що спостерігається.

Гістограму будуватимемо у масштабі відповідної їй функції щільності розподілу (щільності нормального розподілу). Для цього визначимо висоту гістограми як відношення відносної частоти на певному інтервалі до кроку між інтервалами.

За виконаними розрахунками побудуємо графік емпіричного та теоретичного розподілів, результат побудови наведено на рисунку 2.1.

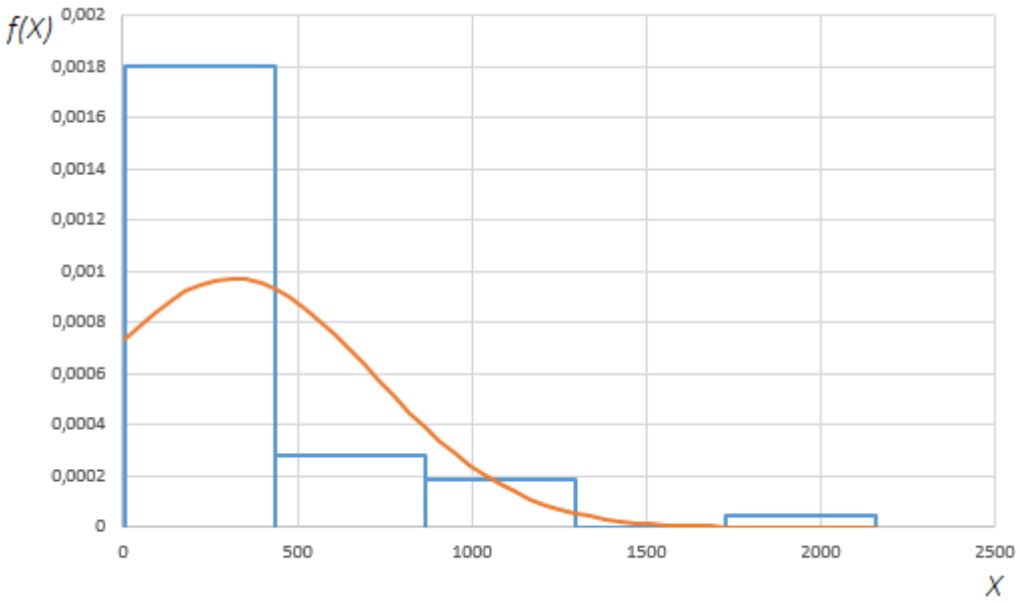

Рисунок 2.1 – Графік теоретичного та емпіричного розподілів X

Перевіримо нульову гіпотезу про нормальність розподілу значень незалежної змінної X. Відповідне значення $\chi^2_{\text{кр}}$, за формулою (2.6): $\chi^2_{\text{кр}} = \chi^2_{\text{кр}}(0{,}05;2) = 5{,}99$.

В результаті розрахунків отримаємо $\chi^2_{\text{спост}} = 40{,}071$. Очевидно, що $\chi^2_{\text{спост}} \geq \chi^2_{\text{кр}}$, тому на рівні значимості 0,05 гіпотезу про нормальність закону розподілу значень незалежної змінної X відхилено. Аналогічно, після перевірки нульової гіпотези про нормальність розподілу значень залежної змінної Y гіпотезу про нормальність закону розподілу значень залежної змінної Y відхилено.

Отримані дані не є нормальними, тому виконаємо їх нормалізацію за допомогою нормалізуючого перетворення $\log 10$. Результат нормалізації даних наведено у таблиці 2.2.

Таблиця 2.2 – Нормалізовані дані

Номер проекту	Z_X	Z_Y	Номер проекту	Z_X	Z_Y
1	0,7781	-0,2211	26	2,1903	0,6853
2	1,1461	-0,1487	27	2,2014	1,0949
3	1,1461	-0,6055	28	2,2480	1,0101
4	1,1760	0,2755	29	2,3655	1,2321
5	1,3010	0,1055	30	2,3892	1,2300
6	1,3424	0,1421	31	2,3945	1,0280
7	1,4771	0,4106	32	2,4472	1,3291
8	1,5185	-0,2684	33	2,4683	1,2531
9	1,6334	0,2639	34	2,4829	1,0753
10	1,6434	0,7524	35	2,5237	1,1911
11	1,6989	0,4888	36	2,5635	1,2256
12	1,7075	0,4501	37	2,6191	1,4608
13	1,8573	0,8131	38	2,6191	1,6536
14	1,8976	0,6749	39	2,6599	1,4425
15	1,9190	0,8909	40	2,6785	1,3328
16	1,9242	0,6776	41	2,7731	1,4511
17	1,9493	0,4675	42	2,7875	1,6847
18	1,9912	0,9388	43	2,7938	1,7068
19	2,0253	0,6923	44	2,8848	1,7914
20	2,0863	0,8798	45	2,9609	1,4816
21	2,0863	0,6268	46	3,0111	2,0456
22	2,0934	0,7320	47	3,0741	1,9229
23	2,1205	0,8028	48	3,0938	1,9430
24	2,1430	1,0699	49	3,3345	2,1212
25	2,1522	0,8233			

Визначимо основні статистичні характеристики нормалізованих даних – оцінку математичного сподівання, дисперсії та середньоквадратичного відхилення за формулами (2.1), (2.2) по незалежній (X) та залежній змінній (Y): $\bar{x} = 2{,}171, S_x^2 = 0{,}341,\ \hat{\sigma}_x = 0{,}584, \bar{y} = 0{,}941, S_y^2 = 0{,}393;\ \hat{\sigma}_y = 0{,}627.$

2.2 Перевірка емпіричних даних на викиди

Для розрахунку квадратів відстані Махаланобіса за формулою (2.3), знайдемо коваріаційну матрицю, яка розраховується за формулою (2.4), а також обернену до неї матрицю. В нашому випадку це буде матриця 2x2, на головній діагоналі якої знаходиться дисперсії по змінним *X* та *Y*, а на побічній – зміщена оцінка дисперсії по цим же змінним. Зміщена оцінка дисперсії має вигляд:

$$\frac{(Z_x-\bar{Z}_x)(Z_y-\bar{Z}_y)}{N} = 0{,}3426.$$

Відповідно коваріаційна матриця та обернена до коваріаційної матриці мають вигляд:

$$\begin{pmatrix} 0{,}3343 & 0{,}3426 \\ 0{,}3426 & 0{,}3855 \end{pmatrix}$$

$$\begin{pmatrix} 33{,}5576 & -29{,}8236 \\ -29{,}8236 & 29{,}0987 \end{pmatrix}$$

Перевірку викидів будемо виконувати за двома критеріями: критеріям Пірсона та критерієм Фішера. Для можливості перевірки викидів за критерієм Фішера розрахуємо статистику за формулою (2.5). Результат розрахунку квадратів відстані Махаланобіса та статистики наведено у таблиці 2.3:

Таблиця 2.3 – Розрахунки квадратів відстані Махаланобіса та статистики для *N*=49

Номер проекту	Z_X	Z_X	d_i^2	T_{S_i}
1	2	3	4	5
1	0,7782	-0,2211	6,8076	3,2662
2	1,1461	-0,1487	3,1532	1,5129
3	1,1461	-0,6055	8,4900	4,0735
4	1,1761	0,2755	5,1386	2,4655
5	1,3010	0,1055	2,2462	1,0777
6	1,3424	0,1421	2,0332	0,9755
7	1,4771	0,4106	1,9661	0,9433
8	1,5185	-0,2684	7,4548	3,5767

Продовження табл. 2.3

1	2	3	4	5
9	1,6335	0,2639	1,2298	0,5901
10	1,6435	0,7524	3,1568	1,5146
11	1,6990	0,4888	0,6621	0,3177
12	1,7076	0,4501	0,6430	0,3085
13	1,8573	0,8131	0,9914	0,4757
14	1,8976	0,6749	0,2203	0,1057
15	1,9191	0,8909	1,0128	0,4859
16	1,9243	0,6776	0,1836	0,0881
17	1,9494	0,4675	1,4191	0,6809
18	1,9912	0,9388	0,7336	0,3520
19	2,0253	0,6923	0,2753	0,1321
20	2,0864	0,8798	0,0321	0,0154
21	2,0864	0,6268	1,0848	0,5205
22	2,0934	0,7320	0,3664	0,1758
23	2,1206	0,8028	0,1633	0,0783
24	2,1430	1,0699	0,4975	0,2387
25	2,1523	0,8233	0,1999	0,0959
26	2,1903	0,6853	1,5321	0,7351
27	2,2014	1,0949	0,3082	0,1479
28	2,2480	1,0101	0,0188	0,0090
29	2,3655	1,2321	0,2946	0,1413
30	2,3892	1,2300	0,2372	0,1138
31	2,3945	1,0280	0,5291	0,2538
32	2,4472	1,3291	0,4698	0,2254
33	2,4683	1,2531	0,2626	0,1260
34	2,4829	1,0753	0,9325	0,4474
35	2,5237	1,1911	0,5843	0,2804
36	2,5635	1,2256	0,6947	0,3333
37	2,6191	1,4608	0,6858	0,3291
38	2,6191	1,6536	1,9768	0,9485
39	2,6599	1,4425	0,7025	0,3370
40	2,6785	1,3328	1,0363	0,4972
41	2,7731	1,4511	1,2431	0,5964
42	2,7875	1,6847	1,4259	0,6841
43	2,7938	1,7068	1,5330	0,7355
44	2,8848	1,7914	1,8520	0,8886
45	2,9609	1,4816	3,1181	1,4961
46	3,0111	2,0456	3,4394	1,6502
47	3,0741	1,9229	2,5025	1,2007
48	3,0938	1,9430	2,6090	1,2518
49	3,3345	2,1212	3,9689	1,9043

Виконаємо порівняння отриманих значень з критичними. При розмірі вибірки квадрат відстані Махаланобіса поводиться як квантіль розподілу χ^2, тобто $\chi^2_{\mathrm{m},\alpha} = \chi^2_{2,0.05} = 5{,}991$.

Бачимо, усі значення квадратів відстані Махаланобіса менші, ніж відповідне значення критерія хі квадрат. Застосуємо критерій Фішера. Його критичне значення $F_{k,N-k,\alpha} = F_{2,47,0.05} = 3{,}195.$

Порівнюючи розраховані значення з критичними, бачимо, що за критерієм χ^2 метрики для 1, 3, 4 та 8 проектів слід вважати викидами, аналогічно за критерієм Фішера - 1, 3 та 8 проектів. Відкинемо 4 проекти та повторимо процес пошуку викидів ще раз.

Для 45 пар даних виконаємо повторний розрахунок основних статистичних характеристик нормалізованих даних: $\bar{x} = 2{,}261$, $S_x^2 = 0{,}263$, $\hat{\sigma}_x = 0{,}513$, $\bar{y} = 1{,}043$, $S_y^2 = 0{,}29$; $\hat{\sigma}_y = 0{,}538$.

У випадку, якщо загальне середнє значення невідоме і розраховується як середнє значення вибірки, тоді дисперсія вибірки буде зміщеною оцінкою. Зміщена оцінка дисперсія розраховується як середнє значення квадратичних відхилень від вибіркового середнього, ділене на число N і має вигляд:

$$\frac{(Z_x - \bar{Z}_x)(Z_y - \bar{Z}_y)}{N} = 0{,}2604.$$

Відповідно коваріаційна матриця та обернена до коваріаційної матриці мають вигляд:

$$\begin{pmatrix} 0{,}2579 & 0{,}2604 \\ 0{,}2604 & 0{,}2838 \end{pmatrix}$$

$$\begin{pmatrix} 52{,}5336 & -48{,}1972 \\ -48{,}1972 & 47{,}7417 \end{pmatrix}$$

Результат розрахунку квадратів відстані Махаланобіса та статистики наведено у таблиці 2.4:

Таблиця 2.4 – Розрахунки квадратів відстані Махаланобіса та статистики для N=45

Номер проекту	Z_X	Z_X	d_i^2	T_{S_i}
1	2	3	4	5
2	1,1461	-0,1487	4,9642	2,3729
5	1,3010	0,1055	3,4987	1,6724

Продовження табл. 2.4

1	2	3	4	5
6	1,3424	0,1421	3,2018	1,5305
7	1,4771	0,4106	2,9332	1,4021
9	1,6335	0,2639	2,2606	1,0806
10	1,6435	0,7524	4,5091	2,1554
11	1,6990	0,4888	1,1988	0,5731
12	1,7076	0,4501	1,2275	0,5868
13	1,8573	0,8131	1,4745	0,7049
14	1,8976	0,6749	0,5041	0,2410
15	1,9191	0,8909	1,4789	0,7069
16	1,9243	0,6776	0,4633	0,2215
17	1,9494	0,4675	2,5274	1,2081
18	1,9912	0,9388	1,0681	0,5106
19	2,0253	0,6923	0,6297	0,3010
20	2,0864	0,8798	0,1187	0,0567
21	2,0864	0,6268	1,9039	0,9101
22	2,0934	0,7320	0,7438	0,3556
23	2,1206	0,8028	0,3848	0,1839
24	2,1430	1,0699	0,6724	0,3214
25	2,1523	0,8233	0,4247	0,2030
26	2,1903	0,6853	2,5120	1,2008
27	2,2014	1,0949	0,3833	0,1832
28	2,2480	1,0101	0,0126	0,0060
29	2,3655	1,2321	0,2636	0,1260
30	2,3892	1,2300	0,1723	0,0823
31	2,3945	1,0280	0,7144	0,3415
32	2,4472	1,3291	0,4466	0,2135
33	2,4683	1,2531	0,1633	0,0781
34	2,4829	1,0753	1,2335	0,5896
35	2,5237	1,1911	0,6325	0,3023
36	2,5635	1,2256	0,7503	0,3586
37	2,6191	1,4608	0,6098	0,2915
38	2,6191	1,6536	2,4751	1,1831
39	2,6599	1,4425	0,6031	0,2883
40	2,6785	1,3328	1,1095	0,5304
41	2,7731	1,4511	1,2772	0,6105
42	2,7875	1,6847	1,4979	0,7160
43	2,7938	1,7068	1,6474	0,7875
44	2,8848	1,7914	2,0035	0,9577
45	2,9609	1,4816	3,7689	1,8016
46	3,0111	2,0456	4,1989	2,0071
47	3,0741	1,9229	2,6836	1,2828
48	3,0938	1,9430	2,8098	1,3431
49	3,3345	2,1212	4,3753	2,0914

Виконаємо порівняння отриманих значень з критичними. $\chi^2_{m,\alpha} = \chi^2_{2,0.05} = 5{,}991$. Бачимо, усі значення квадратів відстані Махаланобіса менші, ніж відповідне значення критерія хі квадрат. Застосуємо критерій Фішера. Його критичне значення $F_{k,N-k,\alpha} = 3{,}214$.

Порівнюючи розраховані значення з критичними, бачимо, що за критерієм χ^2 та за критерієм Фішера метрики для 10 проекту слід вважати викидом. Для 44 пар даних виконаємо повторні розрахунки квадратів відстані Махаланобіса та статистики - розраховані значення статистик менше, ніж критичні, отже усі викиди усунені.

2.3 Побудова нелінійної регресійної моделі для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, та перевірка її якості

Лінійне рівняння регресії визначає залежність оцінки умовного математичного сподівання $\hat{Y}$ залежної випадкової величини Y від k факторів (незалежних змінних) $X_1, X_2, \dots, X_k$.

$$\hat{Y} = b_0 + b_1 X_1 + b_2 X_2 + \ \dots + b_k X_k, \tag{2.7}$$

де $b_0, b_1, b_2, \dots, b_k$ – параметри лінійного регресійного рівняння, оцінки яких як правило знаходять за методом найменших квадратів за емпіричними даними шляхом рішення наступної задачі математичного програмування.

$$\hat{\theta} = argmin\left\{\sum_1^N \left[Y_i - \hat{Y}_i\right]^2\right\}, \tag{2.8}$$

θ – вектор параметрів, що потребують оцінювання,

$\theta = \{b_0, b_1, \dots, b_k\}$;

$\hat{\theta} -$ вектор оцінок параметрів,

$\hat{\theta} = \{\hat{b}_0, \hat{b}_1, \dots, \hat{b}_k\}$;

$N -$ кількість точок емпіричних даних;

$Y_i -$ i-те значення випадкової величини Y;

$\hat{Y}_i - i -$тезначення оцінки $\hat{Y}$,

$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_{1i} + \hat{b}_2 X_{2i} + \cdots + \hat{b}_k X_{ki}$.

Рівняння (2.7) називають множинним (багатофакторним) лінійним рівнянням регресії. Для оцінювання його параметрів замість методу найменших квадратів (2.8) можуть використовуватися інші методи, наприклад, метод максимальної правдоподібності. У разі однофакторного лінійного рівняння регресії

$$\hat{Y} = b_0 + b_1 X_1 \tag{2.9}$$

знаходження оцінок параметрів рівняння регресії (2.9) за методом найменших квадратів (2.8) можна спростити і обчислювати за наступними формулами:

$$\begin{aligned} \hat{b}_1 &= \frac{\sum_{i=1}^{N}(Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{N}(X_{1i} - \overline{X_1})^2}, \\ \hat{b}_0 &= \bar{Y} - \hat{b}_1 \bar{X} \end{aligned} \tag{2.10}$$

де $\bar{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$,

$\bar{X}_1 = \frac{1}{N}\sum_{i=1}^{N} X_{1i}$.

Якість регресійного рівняння та регресійної моделі, як правило, перевіряють за коефіцієнтом детермінації R^2 (The Coefficient of Determination), середньою величиною відносної помилки *MMRE* (Mean Magnitude of Relative Error) і відсотком прогнозованих результатів *PRED* (Percentage of Prediction), для яких величини відносної помилки *MRE* (Magnitude of Relative Error) менші

за 0,25, *PRED*(0,25). Ці показники зазвичай використовуються для оцінювання якості прогнозування за допомогою регресійних моделей і в інженерії програмного забезпечення.

Вибірковий коефіцієнт детермінації R^2 визначається як:

$$R^2 = 1 - \frac{SS_E}{SS_T}, \tag{2.11}$$

де SS_E – сума квадратів залишків (the residual sum of squares) або сума квадратів помилок (the error sum of squares);

SS_T – сума квадратів залишків (the residual sum of squares) або сума квадратів помилок (the error sum of squares);

Нагадаємо, що $SS_T = SS_R + SS_E$.Тут SS_R – сума квадратів регресії (the regression sum of squares).

Коефіцієнт детермінації R^2 інтерпретується як частка мінливості у спостережуваній залежній змінній *Y*, що пояснюється моделлю лінійної регресії. Значення R^2 вказує наскільки емпіричні дані підтверджують математичну модель або, чи краща наша модель за вибіркове середнє. З формули (2.11) ми бачимо, $0 \leq R^2 \leq 1$. Значення R^2 вважається прийнятним, якщо воно більше за 0,75. Велике значення R^2 свідчить про те, що модель успішно пояснює мінливість *Y*. Коли R^2 малий, це може свідчити про те, що потрібно знайти альтернативну модель, яка може врахувати більшу мінливість *Y*. Значення величини відносної похибки *MRE* обчислюється як:

$$MRE_i = \left|\frac{(Y_i - \hat{Y}_i)}{Y_i}\right|. \tag{2.12}$$

Середня величина відносної похибки *MMRE* визначається як:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i. \tag{2.13}$$

Зазвичай $MMRE \leq 0{,}25$ вважається прийнятною точністю моделі. Значення $PRED(0{,}25)$ обчислюється як:

$$PRED(0{,}25) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 \text{ if } MRE_i \leq 0{,}25\\ 0 \qquad \text{otherwise}\end{cases} \tag{2.14}$$

Зазвичай $PRED(0{,}25)$>0,75 вважається прийнятною точністю прогнозування за допомогою регресійних моделей.

Довірчий інтервал (confidence interval) регресії – це такий інтервал, в якому із заданою довірчою імовірністю можна чекати значення оцінки умовного математичного сподівання $\hat{Y}$ залежної випадкової величини Y або більш стисле визначення – це інтервальна оцінка $\hat{Y}$. Фактично довірчий інтервал визначає точність оцінки $\hat{Y}$ за рівнянням регресії.

Інтервал передбачення (prediction interval) регресії – це такий інтервал, в якому із заданою довірчою імовірністю можна чекати на появу значення залежної випадкової величини Y. Фактично інтервал передбачення регресії визначає той інтервал, в якому із заданою довірчою імовірністю знаходяться значення залежної випадкової величини Y, і вихід за який вважається викидом.

Довірчий інтервал лінійної регресії визначається як:

$$\widehat{Y_i} \pm t_{\alpha/2,v} S_Y \left\{\frac{1}{N} + (x^+)^T[(X^+)^T X^+]^{-1}(x^+)\right\}^{1/2}, \tag{2.15}$$

а інтервал передбачення лінійної регресії – як:

$$\widehat{Y_i} \pm t_{\alpha/2,v} S_Y \left\{1 + \frac{1}{N} + (x^+)^T[(X^+)^T X^+]^{-1}(x^+)\right\}^{1/2}, \tag{2.16}$$

де $\widehat{Y_i}$ – результат передбачення за лінійним рівнянням регресії (2.7),

X^+ – матриця центрованих факторів (регресорів), яка містить значення $X_{1i} - \bar{X}_1, X_{2i} - \bar{X}_2, \ldots, X_{ki} - \bar{X}_k$;

$S_Y^2 = \frac{1}{v}\sum_{i=1}^{N}\left(Y_i - \widehat{Y_i}\right)^2$;

$v = N - k - 1$;

$(X^+)^T X^+$ – $k \times k$ матриця.

$$(X^+)^T X^+ = \begin{pmatrix} S_{X_1X_1} & S_{X_1X_2} & \ldots & S_{X_1X_k} \\ S_{X_1X_2} & S_{X_2X_2} & \ldots & S_{X_2X_k} \\ \ldots & \ldots & \ldots & \ldots \\ S_{X_1X_k} & S_{X_2X_k} & \ldots & S_{X_kX_k} \end{pmatrix}$$

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

$q, r = 1,2, \ldots, k.$

У разі однофакторної лінійної регресії (2.9) довірчий інтервал (2.15) можна записати як:

$$\widehat{Y_i} \pm t_{\alpha/2,N-2} S_Y \sqrt{\frac{1}{N} + \frac{(X_1 - \bar{X}_i)^2}{S_{X1X1}}}, \qquad (2.17)$$

а інтервал передбачення (2.16) – як:

$$\widehat{Y_i} \pm t_{\alpha/2,N-2} S_Y \sqrt{1 + \frac{1}{N} + \frac{(X_1 - \bar{X}_i)^2}{S_{X1X1}}}, \qquad (2.18)$$

де $t_{\alpha/2,N-2}$ – квантіль t-розподілу Стьюдента з $N-2$ ступенями свободи та $\alpha/2$ рівнем значущості;

$S_Y^2 = \frac{1}{N-2}\sum_{i=1}^{N}\left(Y_i - \widehat{Y_i}\right)^2$;

$\bar{X}_1 = \frac{1}{N}\sum_{i=1}^{N} X_{1i}$;

$S_{X1X1} = \sum_{i=1}^{N}(X_{1i} - \bar{X}_1)^2.$

В (2.15) та (2.16) значення $\widehat{Y}_i$ визначається за однофакторним лінійним рівнянням регресії (2.9) в залежності від значення фактору X_{1_i}.

Зауважимо, що формули (2.15) - (2.18) побудовані виходячи з припущення нормальності закону розподілу залежної величини *Y*. У разі коли закон розподілу не є нормальним, ці формули як правило дають хибні результати.

Існують чотири основні припущення, які обґрунтовують використання лінійних регресійних моделей, одним з яких є нормальність розподілу похибок, отже, нормальність залежної змінної *Y*. Але для реальних даних це припущення виконується лише в поодиноких випадках. Що призводить до необхідності побудови нелінійних регресійних моделей.

На сьогодні для побудови нелінійних регресійних рівнянь та моделей використовуються методи на основі простого перебору, лінеаризуючих та нормалізуючих перетворень. Для уникнення простого перебору при побудові нелінійних регресійних рівнянь використовують методи, що базуються на нормалізуючих взаємозворотних перетвореннях як одновимірних, так і багатовимірних. Методи на основі нормалізуючих перетворень не потребують використання процедури простого перебору, а, по-друге, при застосуванні бієктивних (bijective) перетворень не відбувається втрата частини інформації як, наприклад, при лінеаризації.

Перш ніж скористатися відповідним методом для побудови нелінійних регресійних рівнянь на основі нормалізуючих перетворень потрібно визначитися з таким перетворенням. Нехай існує взаємозворотне нормалізуюче перетворення негаусівського випадкового вектору $P = \{Y, X_1, X_2, \dots, X_k\}^T$ у гаусівський випадковий вектор $T = \{Z_Y, Z_1, Z_2, \dots, Z_k\}^T$, яке задається як:

$$T = \psi(P), \tag{2.19}$$

і зворотнє перетворення:

$$P = \psi^{-1}(T),$$

тоді $\psi = \{\psi_Y, \psi_1, \psi_2, \dots, \psi_k\}^T$.

Наведемо приклад побудови однофакторного нелінійного рівняння регресії у разі застосування одновимірного перетворення у вигляді десяткового логарифму до кожної змінної:

$$Z_Y = lg\, Y,$$
$$Z_1 = lg\, X_1.$$

Спочатку емпіричні дані нормалізують за перетворенням (2.19). Далі будується однофакторне лінійне регресійне рівняння для нормалізованих даних:

$$\hat{Z}_Y = b_0 + b_1 Z_1. \quad (2.20)$$

За рівнянням (2.20) отримують однофакторне нелінійне рівняння регресії:

$$\hat{Y} = 10^{\hat{b}_0} X_1^{\hat{b}_1}. \quad (2.21)$$

Рівняння виду (2.21) з відповідними параметрами застосовується у якості відомих моделей в інженерії програмного забезпечення: COCOMO (COnstructive COst MOdel) та ISBSG (International Software Benchmarking Standards Group).

Як було зазначено раніше, формули (2.15) - (2.18) для визначення довірчих інтервалів та інтервалів передбачення лінійних регресій побудовані виходячи з припущення нормальності закону розподілу залежної величини *Y*. У разі коли закон розподілу не є нормальним, ці формули як правило дають хибні результати. Для уникнення цього потрібно переходити до нелінійних регресійних моделей та застосування відповідних методів для побудови довірчих інтервалів та інтервалів передбачення нелінійних регресії.

Довірчий інтервал множинної нелінійної регресії визначається як:

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\alpha/2,v}\, S_{Z_Y}\left\{\frac{1}{N} + (z_X^+)^T[(Z_X^+)^T Z_X^+\,]^{-1}(z_X^+)\right\}^{1/2}\right), \qquad (2.22)$$

а інтервал передбачення множинної нелінійної регресії – як:

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\alpha/2,v}\, S_{Z_Y}\left\{1 + \frac{1}{N} + (z_X^+)^T[(Z_X^+)^T Z_X^+\,]^{-1}(z_X^+)\right\}^{1/2}\right).$$

Тут $\hat{Z}_Y$ – результат передбачення за лінійним регресійним рівнянням для нормалізованих даних;

Z_X^+ – матриця центрованих нормалізованих факторів (регресорів), яка містить значення $Z_{1_i} - \bar{Z}_1$, $Z_{2_i} - \bar{Z}_2$, …, $Z_{1_i} - \bar{Z}_1$, $Z_{k_i} - \bar{Z}_k$;

$(Z_X^+)^T Z_X^+$ – $k \times k$ матриця:

$$(Z_X^+)^T Z_X^+ = \begin{pmatrix} S_{Z_1Z_1} & S_{Z_1Z_2} & \dots & S_{Z_1Z_k} \\ S_{Z_1Z_2} & S_{Z_2Z_2} & \dots & S_{Z_2Z_k} \\ \dots & \dots & \dots & \dots \\ S_{Z_1Z_k} & S_{Z_2Z_k} & \dots & S_{Z_kZ_k} \end{pmatrix}$$

де $S_{Z_qZ_r} = \sum_{i=1}^{N}\left[Z_{q_i} - \bar{Z}_q\right]\left[Z_{r_i} - \bar{Z}_r\right]$.

У разі однофакторної нелінійної регресії довірчий інтервал (2.22) можна записати як:

$$\psi_Y^{-1}\left(\hat{Z}_{Y_i} \pm t_{\alpha/2,N-2} S_Z \sqrt{\frac{1}{N} + \frac{(Z_1 - \bar{Z}_i)^2}{S_{Z_1Z_1}}}\right), \qquad (2.23)$$

а інтервал передбачення – як:

$$\psi_Y^{-1}\left(\hat{Z}_{Y_i} \pm t_{\alpha/2,N-2} S_Z \sqrt{1 + \frac{1}{N} + \frac{(Z_1 - \bar{Z}_i)^2}{S_{Z_1Z_1}}}\right), \qquad (2.24)$$

де $t_{\alpha/2,N-2}$ – квантіль t-розподілу Стьюдента з $N-2$ ступенями свободи та $\alpha/2$ рівнем значущості;

$$S_Z^2 = \frac{1}{N-2}\sum_{i=1}^{N}\left(Z_i - \hat{Z}_i\right)^2;$$

$$\bar{Z}_1 = \frac{1}{N}\sum_{i=1}^{N} Z_{1_i};$$

$$S_{Z_1 Z_1} = \sum_{i=1}^{N}\left(Z_{1_i} - \bar{Z}_1\right)^2.$$

В (2.23) та (2.24) значення $\hat{Z}_{Y_i}$ визначається за однофакторним лінійним рівнянням регресії (2.20) для нормалізованих даних в залежності від значення фактору Z_{1_i}.

Для побудови регресійних моделей використовуватимемо дані без викидів, які було отримано у пункті 2.2. За формулою (2.10) визначимо коефіцієнти лінійного рівняння регресії для ненормалізованих даних: $b_0 = -0{,}5102$, $b_1 = 0{,}0677$.

Одним з критеріїв теоретичного обґрунтування використання лінійної регресії є нормальність розподілу відхилень ε. Визначимо значення відхилень між емпіричними даними та рівнянням регресії (лінією регресії). Відносно отриманих значень виконаємо перевірку гіпотези про нормальність розподілу відхилень з достовірчою ймовірністю 0.95.

Визначимо основні статистичні характеристики отриманої вибірки – оцінку математичного сподівання, дисперсії та середньоквадратичного відхилення $\bar{x} = -2{,}3201\text{E}-15$, $S_x^2 = 84{,}9307$, $\hat{\sigma}_x = 9{,}2158$.

Перевіримо нормальність закону розподілу відхилень. Розмах вибірки: $R = 73{,}6725$.

За формулою Старджеса визначимо кількість інтервалів для побудови гістограми емпіричного розподілу $m = log_2 N + 1 = 3{,}3\, lg\, N + 1 = 6{,}472 = 6$.

Оскільки графік щільності ймовірності нормального розподілу є унімодальним, при побудові гістограми слід обрати непарну кількість

інтервалів: $m = 5$. Довжина інтервалу гістограми емпіричного розподілу $h = \frac{R}{m} = \frac{73{,}6725}{5} = 14{,}7345$.

За виконаними розрахунками побудуємо графік емпіричного розподілу значень похибок регресії, результат побудови наведено на рисунку 2.2:

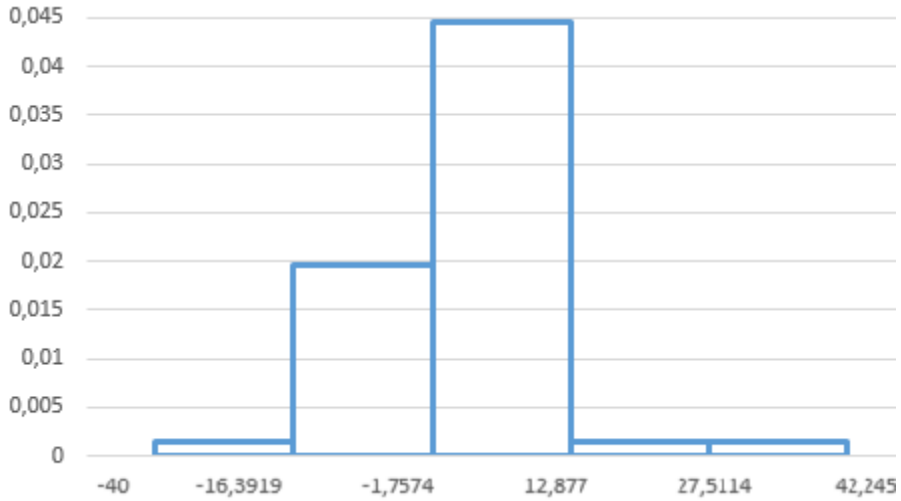

Рисунок 2.2 – Графік емпіричного розподілу

Перевіримо нульову гіпотезу про нормальність розподілу значень незалежної змінної *X*. Відповідне значення $\chi^2_{\text{кр}} = \chi^2_{\text{кр}}(0{,}05{,}2) = 5{,}99$.

В результаті розрахунків отримаємо $\chi^2_{\text{спост}} = 18{,}82$. Очевидно, що $\chi^2_{\text{спост}} \geq \chi^2_{\text{кр}}$, тому на рівні значимості 0,05 гіпотезу про нормальність закону розподілу не прийнято. Прийняття відповідної нульової гіпотези є одним з критеріїв, які дозволяють виконати побудову лінійного рівняння регресії.

Спробуємо знайти більш оптимальну регресійну модель, що описує дані метрик програмного забезпечення. Побудуємо нелінійну регресійну модель на основі взаємозворотнього нормалізуючого перетворення $log\,10$. Нормалізуємо початкові дані з метрик програмного забезпечення.

Визначимо основні статистичні характеристики отриманої вибірки – оцінку математичного сподівання, дисперсії та середньоквадратичного відхилення: $\bar{x} = 2{,}275, S_x^2 = 0{,}2609, \hat{\sigma}_x = 0{,}5108$.

Після нормалізації вихідних використаємо формулу (2.10) для знаходження значень оцінок параметрів b_0, b_1 для побудови лінійного рівняння регресії для нормалізованих даних, отримаємо наступні значення: $b_0 = -1{,}469$, $b_1 = 1{,}226$.

Перевіримо нормальність закону розподілу похибок. Розмах вибірки: $R = 1{,}8915$. Кількість інтервалів на гістограмі $m = 5$. Довжина інтервалу гістограми емпіричного розподілу: $h = \frac{R}{m} = \frac{1{,}8915}{5} = 0{,}3783$.

За виконаними розрахунками побудую графік емпіричного та теоретичного розподілів похибок лінійної регресії для нормалізованих даних, результат побудови наведено на рисунку 2.3:

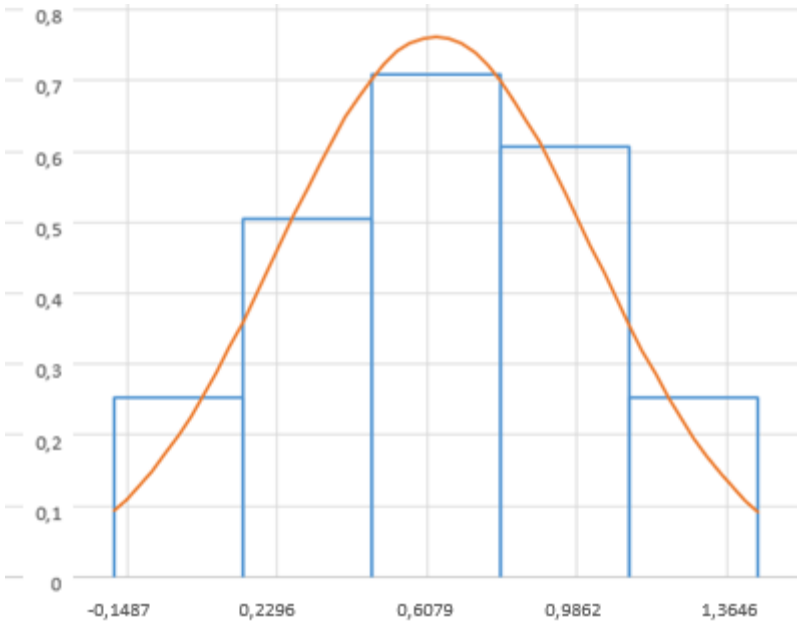

Рисунок 2.3 – Графік теоретичного та емпіричного розподілів похибок

Перевіримо нульову гіпотезу про нормальність розподілу значень незалежної змінної x. Відповідне значення $\chi^2_{\text{кр}} = \chi^2_{\text{кр}}(0{,}05{,}2) = 5{,}99$.

В результаті розрахунків отримаємо $\chi^2_{\text{спост}} = 1{,}47$. Очевидно, що $\chi^2_{\text{спост}} \leq \chi^2_{\text{кр}}$, тому на рівні значимості 0,05 гіпотезу про нормальність закону розподілу прийнято.

За формулою (2.21) виконаємо побудову однофакторного нелінійного рівняння регресії. Визначимо коефіцієнти, які дозволяють оцінити якість побудованої нелінійної регресійної моделі: $R^2 = 0{,}8877$, $MMRE = 0{,}2704$, $PRED(0{,}25) = 0{,}5227$.

Визначимо довірчі інтервали та інтервали передбачення нелінійної регресії для метрик програмного забезпечення для достовірчої ймовірності 0,95 використовуючи формулу (2.24) та (2.25). Для розрахунків знайдемо наступні значення: $t_{\alpha/2,N-2} = 2{,}3246$, $S_Y = 0{,}1401$, $S_{X1X1} = 11{,}2179$, $\bar{X} = 2{,}2753$.

Дані побудованих нелінійної однофакторної регресії, довірчого інтервалу та інтервалу передбачення наведено у таблиці 2.5, де Conf_L та Conf_R – відповідно ліва та права границі довірчого інтервалу, Pred_L та Pred_R – відповідно ліва та права границі інтервалу передбачення:

Таблиця 2.5 – Дані моделювання нелінійної регресії

Номер	NC (X)	KLOC	$\hat{Y}$	Conf_L	Conf_R	Pred_L	Pred_R
1	2	3	4	5	6	7	8
2	14	0,71	0,7738	0,5865	1,0209	0,3044	1,9674
5	20	1,275	1,1167	0,8733	1,4279	0,4535	2,7498
6	22	1,387	1,2317	0,9712	1,5621	0,5024	3,0197
7	30	2,574	1,6944	1,3712	2,0937	0,7126	4,0290
9	43	1,836	2,4535	2,0433	2,9461	1,0151	5,9300
11	50	3,082	2,8651	2,4131	3,4017	1,2150	6,7563
12	51	2,819	2,9240	2,4663	3,4666	1,2349	6,9235
13	72	6,503	4,1685	3,5992	4,8279	1,8241	9,5262
14	79	4,73	4,5858	3,9816	5,2816	1,9812	10,6144
15	83	7,778	4,8247	4,2009	5,5411	2,1255	10,9517
16	84	4,76	4,8845	4,2558	5,6060	2,1108	11,3029
17	89	2,934	5,1837	4,5307	5,9309	2,1933	12,2515
18	98	8,686	5,7235	5,0268	6,5168	2,5316	12,9400
19	106	4,924	6,2045	5,4687	7,0393	2,6850	14,3372
20	122	7,583	7,1695	6,3538	8,0899	3,1555	16,2894
21	122	4,234	7,1695	6,3538	8,0899	3,0831	16,6722
22	124	5,395	7,2904	6,4644	8,2218	3,1667	16,7838
23	132	6,351	7,7745	6,9070	8,7509	3,3989	17,7831
24	139	11,747	8,1987	7,2939	9,2158	3,6640	18,3458

Продовження табл. 2.5

1	2	3	4	5	6	7	8
25	142	6,657	8,3807	7,4596	9,4156	3,6706	19,1351
26	155	4,845	9,1707	8,1768	10,2853	3,9660	21,2056
27	159	12,443	9,4141	8,3971	10,5543	4,2150	21,0263
28	177	10,236	10,5117	9,3859	11,7726	4,6762	23,6297
29	232	17,063	13,8840	12,3769	15,5745	6,2761	30,7139
30	245	16,982	14,6846	13,0769	16,4899	6,6371	32,4895
31	248	10,665	14,8695	13,2381	16,7020	6,6239	33,3793
32	280	21,336	16,8459	14,9487	18,9838	7,6618	37,0387
33	294	17,911	17,7126	15,6925	19,9928	8,0179	39,1298
34	304	11,894	18,3325	16,2222	20,7172	8,1961	41,0048
35	334	15,529	20,1953	17,8034	22,9085	9,1041	44,7986
36	366	16,81	22,1875	19,4783	25,2736	10,0254	49,1040
37	416	28,892	25,3101	22,0734	29,0215	11,5973	55,2374
38	416	45,038	25,3101	22,0734	29,0215	11,7016	54,7449
39	457	27,699	27,8787	24,1838	32,1381	12,7618	60,9021
40	477	21,516	29,1341	25,2081	33,6715	13,2536	64,0423
41	593	28,254	36,4428	31,0912	42,7157	16,6898	79,5743
42	613	48,386	37,7073	32,0966	44,2988	17,4546	81,4594
43	622	50,913	38,2767	32,5483	45,0133	17,7330	82,6204
44	767	61,863	47,4804	39,7665	56,6906	22,0610	102,1889
45	914	30,312	56,8617	46,9876	68,8108	26,0825	123,9626
46	1026	111,063	64,0386	52,4353	78,2094	29,9324	137,0066
47	1186	83,726	74,3292	60,1489	91,8526	34,6607	159,3977
48	1241	87,704	77,8760	62,7838	96,5962	36,3311	166,9279
49	2160	132,195	137,6879	105,8636	179,0789	64,4187	294,2925

Результат побудови наведено на рисунку 2.4.

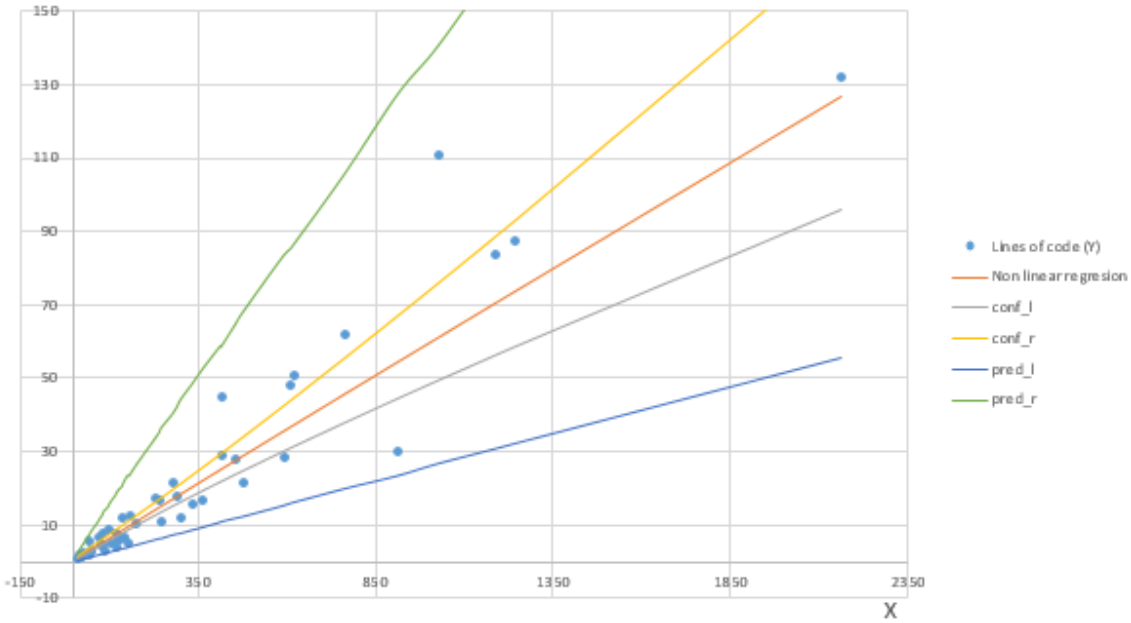

Рисунок 2.4 –Довірчі інтервали та інтервали передбачення нелінійної регресії

Аналізуючи побудовані інтервали, можна побачити, що у випадку нелінійної регресійної моделі жодне значення змінної *Y* не знаходиться поза межами інтервалу передбачення.

Порівняємо побудовану нелінійну регресійну модель з іншою відомою опублікованою моделлю.

В процесі пошуку та аналізу існуючих моделей для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд не було знайдено жодних опублікованих нелінійних регресійних моделей, котрі відповідають темі даній кваліфікаційній роботі, тобто котрі були б побудовані для проектів персональних застосунків-органайзерів для Андроїд чи проектів персональних застосунків на Kotlin. Саме тому для порівняння моделі, побудованої в даній кваліфікаційній роботі, будемо розглядати нелінійні регресійні моделі для проектів веб-додатків, створених мовою Java, як найближчі за тематикою до теми даної роботи. З існуючих моделей було вирішено проводити порівняння з такою моделлю [17].

В результаті розрахунків отримали такі значення параметрів якості порівняльної регресійної моделі: *MMRE*=1,0269, *PRED*(0,25)=0,2045.

Визначимо довірчі інтервали та інтервали передбачення порівняльної моделі для достовірчої ймовірності 0,95 використовуючи формулу (2.24) та (2.25). Отримані дані наведено у таблиці 2.6, де Conf_L та Conf_R – відповідно ліва та права границі довірчого інтервалу, Pred_L та Pred_R – відповідно ліва та права границі інтервалу передбачення:

Таблиця 2.6 – Дані порівняльної нелінійної регресії

Номер	NC (X)	KLOC (Y)	$\hat{Y}$	Conf_L	Conf_R	Pred_L	Pred_R
1	2	3	4	5	6	7	8
2	14	0,71	0,86	0,49	1,53	0,17	4,48
5	20	1,275	1,34	0,81	2,22	0,26	6,79
6	22	1,387	1,51	0,92	2,45	0,30	7,59
7	30	2,574	2,20	1,43	3,40	0,44	10,94
9	43	1,836	3,42	2,35	4,99	0,70	16,76
11	50	3,082	4,12	2,90	5,86	0,85	20,06
12	51	2,819	4,22	2,98	5,99	0,87	20,54
13	72	6,503	6,44	4,77	8,72	1,34	31,04
14	79	4,73	7,22	5,40	9,66	1,50	34,70
15	83	7,778	7,67	5,77	10,20	1,60	36,83
16	84	4,76	7,79	5,87	10,34	1,62	37,37
17	89	2,934	8,36	6,34	11,02	1,74	40,07
18	98	8,686	9,41	7,20	12,28	1,97	45,02
19	106	4,924	10,36	7,99	13,43	2,17	49,50
20	122	7,583	12,31	9,60	15,78	2,58	58,71
21	122	4,234	12,31	9,60	15,78	2,58	58,71
22	124	5,395	12,55	9,80	16,08	2,63	59,89
23	132	6,351	13,55	10,63	17,29	2,84	64,62
24	139	11,747	14,44	11,36	18,37	3,03	68,82
25	142	6,657	14,83	11,67	18,83	3,11	70,63
26	155	4,845	16,51	13,04	20,90	3,47	78,60
27	159	12,443	17,03	13,46	21,54	3,58	81,09
28	177	10,236	19,43	15,39	24,52	4,08	92,46
29	232	17,063	27,07	21,38	34,29	5,68	128,91
30	245	16,982	28,94	22,80	36,74	6,08	137,88
31	248	10,665	29,38	23,13	37,31	6,17	139,96
32	280	21,336	34,09	26,67	43,59	7,15	162,60
33	294	17,911	36,20	28,22	46,43	7,59	172,71
34	304	11,894	37,71	29,33	48,49	7,90	180,02

Продовження табл.2.6

1	2	3	4	5	6	7	8
35	334	15,529	42,33	32,66	54,85	8,86	202,30
36	366	16,81	47,36	36,23	61,89	9,89	226,66
37	416	28,892	55,41	41,82	73,41	11,55	265,84
38	416	45,038	55,41	41,82	73,41	11,55	265,84
39	457	27,699	62,18	46,42	83,29	12,93	298,92
40	477	21,516	65,53	48,67	88,25	13,62	315,36
41	593	28,254	85,59	61,75	118,64	17,68	414,25
42	613	48,386	89,15	64,01	124,15	18,40	431,87
43	622	50,913	90,75	65,03	126,65	18,72	439,85
44	767	61,863	117,35	81,50	168,97	24,05	572,70
45	914	30,312	145,51	98,31	215,38	29,62	714,81
46	1026	111,063	167,68	111,17	252,92	33,97	827,62
47	1186	83,726	200,30	129,62	309,52	40,32	994,97
48	1241	87,704	211,75	135,98	329,74	42,54	1054,08
49	2160	132,195	417,87	243,41	717,36	81,50	2142,62

У випадку порівняльної нелінійної регресійної моделі жодне значення змінної *Y* також не знаходиться поза межами інтервалу передбачення.

Із наведених даних бачимо, що для побудованої нелінійної регресійної моделі отримане значення R^2 є більш ніж задовільним. Значення *MMRE* також є достатньо задовільним і знаходиться у допустимих межах (<0,3). Однак значення *PRED*(0,25) є незадовільним, з чого можна зробити декілька висновків щодо покращення результатів моделювання: потрібно використовувати інший вид нормалізуючого перетворення, наприклад перетворення Джонсона, або будувати багатофакторну модель, якість котрої перевищує відповідну однофакторну модель.

Значення показників якості регресійних моделей покращені для нелінійного рівняння регресії в порівнянні з існуючою моделлю: *MMRE* на 73,66%, *PRED*(0,25) на 155,6%.

Обидві нелінійні регресійні моделі, побудована в даній кваліфікаційній роботі та порівняльна, не мають значень змінної Y, котрі виходили б за межі інтервалу передбачення. Проте, нелінійну регресійну модель, побудовану в

даній кваліфікаційній роботі, вважаємо кращої якості через вужчі довірчі інтервали та інтервали передбачення, ніж у порівняльної моделі.

Отримані результати в цілому показують, що для підвищення достовірності оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Android, доцільно застосовувати нормалізуюче перетворення, нелінійну регресію та викиди регресії, однак треба обрати краще нормалізуюче перетворення, ніж на основі десяткового логарифму та багатофакторну модель.

Регресійна модель, побудована з використанням нормалізуючого перетворення за допомогою десяткового логарифму, може використовуватись для передбачення розміру персональних застосунків-органайзерів для Андроїд.

2.4 Постановка задачі на розробку ПЗ для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд

Виходячи з аналізу існуючих методів і моделей для оцінювання розміру програмного забезпечення та побудувавши нелінійну регресійну модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Android, сформулюємо наступну постановку задачі на розробку програмного забезпечення.

В даній роботі необхідно розробити програмне забезпечення, котре виконуватиме наступні функції:

1) надання можливість вносити дані про проекти персональних застосунків (кількість рядків коду та кількість класів);

2) виконання нормалізації внесених даних за допомогою нормалізуючого перетворення на основі десяткового логарифму;

3) видалення викидів;

4) побудова лінійної регресійної моделі;

5) побудова довірчих інтервалів на основі нормалізованих даних;

6) побудова інтервалів передбачення на основі нормалізованих даних;

7) побудова нелінійної регресійної моделі на основі лінійного рівняння регресії;

8) розрахування метрик якості;

9) оцінювання розміру персональних застосунків.

Детальні вимоги до програмного забезпечення наведені у додатку А – Технічне завдання.

3 ПРОЕКТ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПЕРСОНАЛЬНИХ ЗАСТОСУНКІВ-ОРГАНАЙЗЕРІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN ДЛЯ АНДРОЇД

Для розробки проекту програми, котрий складається з ескізного, технічного та робочого проектів, було обрано об’єктно-орієнтовану методологію та уніфіковану мову моделювання UML.

3.1 Ескізній проект програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд

Для розробки програми «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд» було обрано мову моделювання UML, оскільки дана мова моделювання є об’єктно-орієнтованою та простою в розумінні і користуванні. Уніфікована мова моделювання (unified modeling language) є графічною мовою для візуалізації, специфікування, конструювання та документування систем.

За допомогою UML можна розробити детальний план програми, що розроблюється, який буде включати не тільки її концептуальні елементи, такі як системні функції і бізнес-процеси, але і конкретні особливості, наприклад класи, написані на спеціальних мовах програмування, схеми баз даних і програмні компоненти багаторазового використання. Мова моделювання UML підходить для моделювання будь-якого ПЗ – інформаційних систем до web-додатків і вбудованих систем реального часу. Це дуже виразна мова, призначена для представлення ПЗ з усіх точок зору, що відносяться до його розробки та впровадження. Попри широкий спектр виразних засобів, UML простий для розуміння і застосування. UML є об'єктно-орієнтованою мовою моделювання, в результаті чого методи описання результатів аналізу та

проектування ПЗ семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах програмування [18].

Для розробки програми «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд» на етапі ескізного проекту було виявлено основних акторів та варіанти використання ПЗ, та розроблено специфікації виявлених варіантів використання. Використовуючи мову моделювання UML було побудовано діаграму варіантів використання та концептуальну модель програмного забезпечення, також було розроблено діаграми діяльності для основних варіантів використання. Згідно зі специфікованими варіантами використання та побудованими сценаріями, представленими у вигляді діаграм діяльності, було розроблено ескіз користувацького інтерфейсу.

3.1.1 Побудова моделі варіантів використання

Для визначення основних вимог до розробки програмного продукту використовують діаграму варіантів використання, котра описує функціональність системи у вигляді варіантів використання-прецедентів, а також діючих осіб-акторів, що взаємодіють з системою. Діаграма варіантів використання дозволяє отримати уявлення про функціональну поведінку системи та її взаємодію з користувачем.

На основі проведеного аналізу та постановки задачі до розробки ПЗ, можемо виявити акторів та варіанти використання ПЗ, що розроблюється. Програма для оцінювання розміру персональних застосунків-органайзерів, що створюється мовою Kotlin для Android, яка моделюється у даному розділі, передбачає наявність однієї діючої особи – користувача ПЗ, котрий буде повноцінно використовувати всі функції програмного продукту.

Короткий опис виявлених акторів програми представлено в табл. 3.1.

Таблиця 3.1 – Короткий опис виявлених акторів

Актор	Короткий опис
Користувач	Діюча особа, котра взаємодіє з програмним продуктом та має наступні можливості: введення вхідних даних, нормалізація введених даних, побудова лінійної та нелінійної регресійної моделі, побудова довірчих інтервалів та інтервалів передбачення, оцінювання якості побудованих регресійних моделей, оцінювання розміру програмного проекту

Короткий опис виявлених варіантів використання, котрі відображають функції програми, що розроблюється, наведено в табл. 3.2.

Таблиця 3.2 – Опис варіантів використання програми

Назва варіанту використання	Короткий опис
Введення вхідних даних	Варіант використання дозволяє користувачу ПЗ ввести вхідні дані з обраного ним файлу
Нормалізація вхідних даних	Варіант використання надає користувачу можливість нормалізувати вхідні дані за допомогою нормалізуючого перетворення на основі десяткового логарифму
Побудова лінійної регресії	Варіант використання надає користувачу можливість побудувати лінійну регресійну модель на основі нормалізованих вхідних даних
Побудова нелінійної регресії	Варіант використання надає користувачу можливість побудувати нелінійну регресійну модель на основі нормалізованих вхідних даних
Побудова довірчих інтервалів	Варіант використання надає користувачу можливість побудувати довірчі інтервали регресійної моделі
Побудова інтервалів передбачення	Варіант використання надає користувачу можливість побудувати інтервали передбачення регресійної моделі
Оцінювання якості регресійної моделі	Варіант використання надає користувачу можливість розрахувати показники якості регресійної моделі: R^2, *MMRE, PRED*(0,25)
Оцінювання розміру	Варіант використання надає користувачу можливість виконувати оцінювання розміру ПЗ за допомогою нелінійної регресії

На основі виявлених акторів та варіантів використання системи побудуємо модель варіантів використання. Розроблену діаграму варіантів використання для програми «Нелінійна регресійна модель для оцінювання

розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд» представлено на рисунку 3.1.

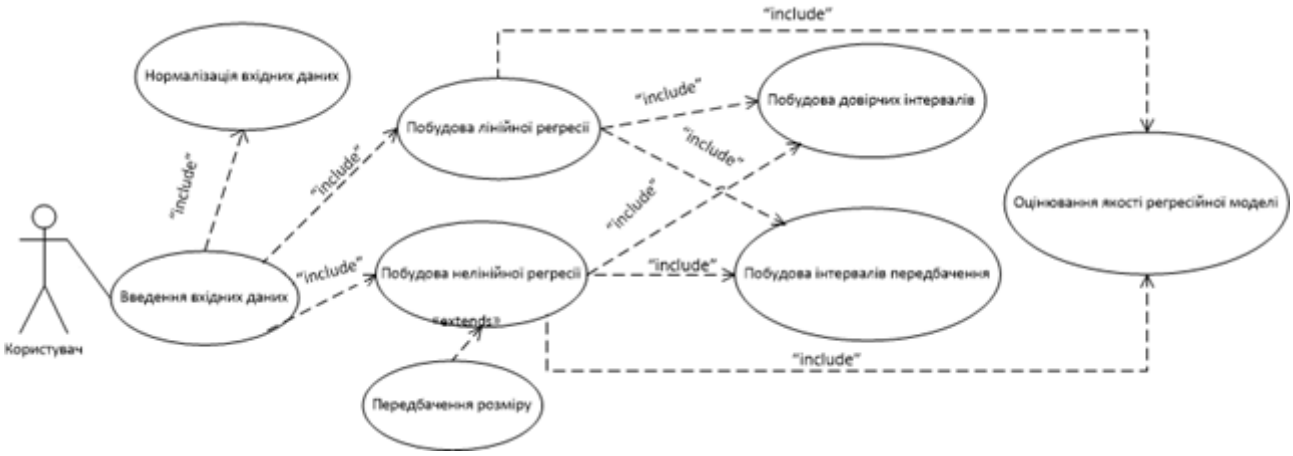

Рисунок 3.1 – Діаграма варіантів використання програми

Таким чином, побудувавши діаграму варіантів використання ПЗ для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Android, маємо основу для подальшого проектування ПЗ. За допомогою спроектованої моделі варіантів використання розробимо специфікації виявлених варіантів використання для формалізації функціональних вимог до програми, що розробляється.

3.1.2 Специфікації варіантів використання

Конкретизуємо основні варіанти використання, змодельовані у попередньому пункті 3.1.1.

1) Назва варіанту використання: Введення вхідних даних

Основна діюча особа: Користувач

Короткий опис: Варіант використання «Введення вхідних даних» дозволяє користувачу ПЗ отримати вхідні дані з обраного ним файлу.

Потоки подій

Основний потік: Користувач обирає шлях до файлу з вхідними даними. Система зчитує вхідні дані з обраного файлу та аналізує їх.

Альтернативі потоки: Якщо обраний файл не існує, є неправильного формату, є пустим чи зберігає дані неправильного формату, то буде виведено повідомлення про помилку та пропозиція повторити внесення даних.

Передумови: Користувач має запустити програмний продукт та натиснути на кнопку для введення вхідних даних.

Постумови: Система надає користувачу можливість виконувати варіанти використання «Нормалізація вхідних даних».

2) Назва варіанту використання: Нормалізація вхідних даних

Основна діюча особа: Користувач

Короткий опис: Варіант використання «Нормалізація вхідних даних» надає користувачу можливість виконує нормалізацію даних для стовпчиків «Y», «X», використовуючи нормалізуюче перетворення нна основі десяткового логарифму.

Потоки подій

Основний потік: Система отримує дані з обраного користувачем файлу під час виконання варіанту використання «Введення вхідних даних» та здійснює їх нормалізацію на основі десяткового логарифму.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Введення вхідних даних».

Постумови: Система надає користувачу можливість виконувати варіант використання «Побудова нелінійної регресії» та «Побудова лінійної регресії».

Спеціальні умови: Не визначені.

3) Назва варіанту використання: Побудова лінійної регресії

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість побудувати лінійну регресійну модель.

Потоки подій

Основний потік: Система розраховує необхідні коефіцієнти для лінійного рівняння регресії на основі даних, отриманих в результаті виконання користувачем варіанту використання «Нормалізація вхідних даних», та виконує побудову лінійної регресійної моделі.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Нормалізація вхідних даних».

Постумови: Виведення побудованої лінійної регресійної моделі на екран. Система надає користувачу можливість виконувати варіанти використання «Оцінювання розміру», «Побудова довірчих інтервалів» та «Побудова інтервалів передбачення».

4) Назва варіанту використання: Побудова нелінійної регресії

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість побудувати нелінійну регресійну модель.

Потоки подій

Основний потік: Система розраховує необхідні коефіцієнти для нелінійного рівняння регресії на основі даних, отриманих в результаті виконання користувачем варіанту використання «Нормалізація вхідних даних», та виконує побудову нелінійної регресійної моделі.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Нормалізація вхідних даних».

Постумови: Виведення побудованої нелінійної регресійної моделі на екран. Система надає користувачу можливість виконувати варіанти використання «Оцінювання розміру», «Побудова довірчих інтервалів» та «Побудова інтервалів передбачення».

5) Назва варіанту використання: Побудова довірчих інтервалів

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість побудувати довірчі інтервали регресійної моделі

Потоки подій

Основний потік: Користувач обирає регресійну модель, для якої потрібно побудувати довірчі інтервали. Система здійснює розрахунки необхідних значень для побудови та виконує побудову довірчих інтервалів для обраної користувачем регресійної моделі.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Побудова лінійної регресії» або «Побудова нелінійної регресії»

Постумови: Виведення побудованих довірчих інтервалів та регресійної моделі на екран.

6) Назва варіанту використання: Побудова інтервалів передбачення

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість побудувати інтервали передбачення обраної регресійної моделі.

Потоки подій

Основний потік: Користувач обирає регресійну модель, для якої потрібно побудувати інтервали передбачення. Система здійснює розрахунки необхідних значень для побудови та виконує побудову інтервалів передбачення для обраної користувачем регресійної моделі.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Побудова лінійної регресії» або «Побудова нелінійної регресії».

Постумови: Виведення побудованих інтервалів передбачення та регресійної моделі на екран.

7) Назва варіанту використання: Оцінювання якості регресійної моделі

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість розрахувати показники якості регресійної моделі: R^2, *MMRE, PRED*(0,25).

Потоки подій

Основний потік: Користувач обирає регресійну модель, для якої потрібно розрахувати значення показників якості. Система розраховує показники якості обраної користувачем регресійної моделі, побудованої внаслідок виконання користувачем варіанту використання «Побудова лінійної регресії» або «Побудова нелінійної регресії»: R^2, *MMRE, PRED*(0,25).

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Побудова лінійної регресії» або «Побудова нелінійної регресії».

Постумови: Виведення розрахованих показників якості регресійної моделі на екран.

8) Назва варіанту використання: Оцінювання розміру

Основна діюча особа: Користувач

Короткий опис: Варіант використання надає користувачу можливість виконувати оцінювання розміру ПЗ за допомогою нелінійної регресії.

Потоки подій

Основний потік: Система виконує оцінювання розміру ПЗ: розраховує показники якості нелінійної регресійної моделі, розраховує та будує інтервали передбачення та довірчі інтервали для нелінійної регресійної моделі.

Альтернативні потоки: Відсутні.

Передумови: Користувач має виконати варіант використання «Побудова нелінійної регресії».

Постумови: Виведення результатів оцінювання на екран.

3.1.3 Побудова діаграм діяльності для основних варіантів використання

Представлення поведінки системи відображається у зміні станів системи в процесі роботи (в залежності від часу). Для того щоб це зобразити, зазвичай використовують діаграму станів чи діаграму активності.

Діаграма діяльності представляє собою особливу форму кінцевого автомата, в якій показуються процес обчислень і потоки робіт. У ній виділяються не звичайні стани об'єкта, а стани виконуваних обчислень - стани дій. При цьому потрібно, щоб процес обчислень не переривався зовнішніми подіями. Тобто, діаграми діяльності дуже схожі на блок-схеми алгоритмів. Діаграма схем станів - одна з діаграм, що моделюють динаміку систем. Діаграма схем станів відображає кінцевий автомат, виділяючи потік управління, наступний від стану до стану [19].

Для достовірнішого відображення послідовності дій, що здійснюються у процесі виконання системою основних варіантів використання, необхідно створити діаграми діяльності для них. Діаграми діяльності було розроблено для двох основних варіантів використання – «Побудова лінійної регресії» та «Оцінювання розміру», специфікованих в пункті 3.1.2.

Для варіанту використання «Побудова лінійної регресії» було розроблено діаграму діяльності, що представлена на рисунку 3.2.

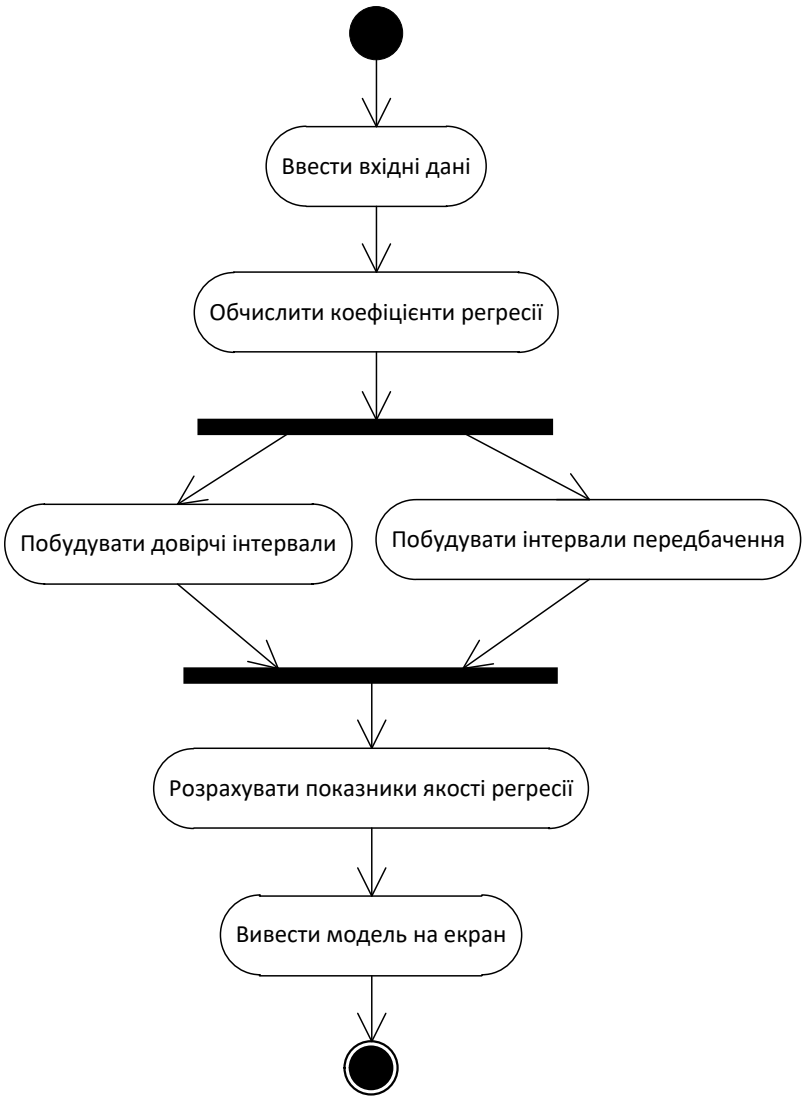

Рисунок 3.2 – Діаграма діяльності «Побудова лінійної регресії»

Для варіанту використання «Оцінювання розміру» було розроблено діаграму діяльності, що представлена на рисунку 3.3.

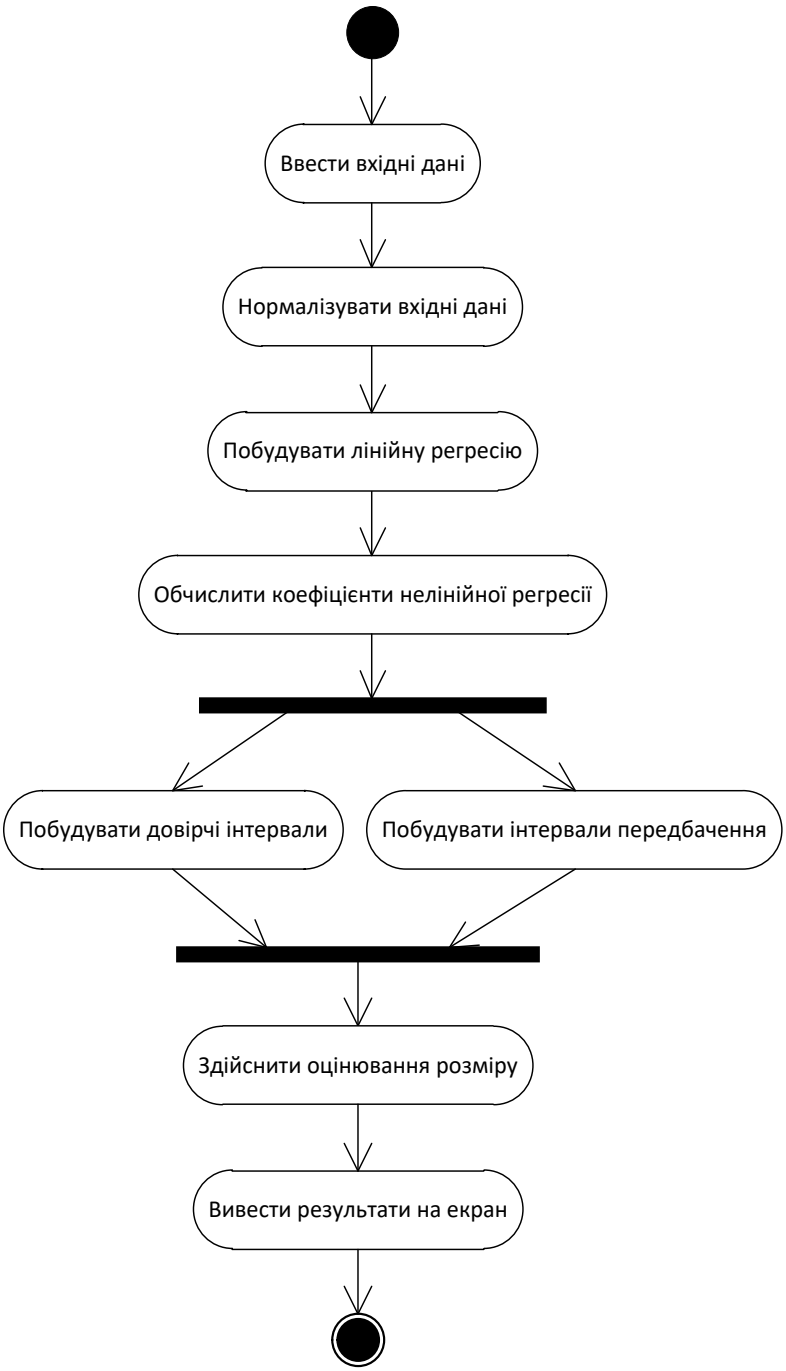

Рисунок 3.3 – Діаграма діяльності «Передбачення розміру»

Отже, було розроблено діаграми діяльності двох основних варіантів використання – «Побудова лінійної регресії» та «Оцінювання розміру» для формалізації сценаріїв ПЗ.

3.1.4 Побудова концептуальної моделі

Концептуальне проектування - побудова змістовної моделі предметної галузі. Модель предметної галузі, що складається з переліку сутностей, які використовуються для опису цієї галузі, разом з їх властивостями, характеристиками та взаємозв'язками є концептуальною моделлю. Подібна модель створюється без орієнтації на модель СКБД та модель даних. Для створення концептуальної моделі можна використати діаграму «Сутність-Зв’язок» (Entity Relation, ER) або контекстну діаграму потоків даних. Концептуальна модель включає в себе опис понять предметної галузі та зв’язків між ними [20].

На основі аналізу вимог до програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, необхідно розробити концептуальну модель даних, яка відображатиме основні сутності ПЗ, разом з їх властивостями та взаємозв'язками.

Концептуальна модель програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, представлено у вигляді діаграму «Сутність-Зв’язок» на рисунку 3.4.

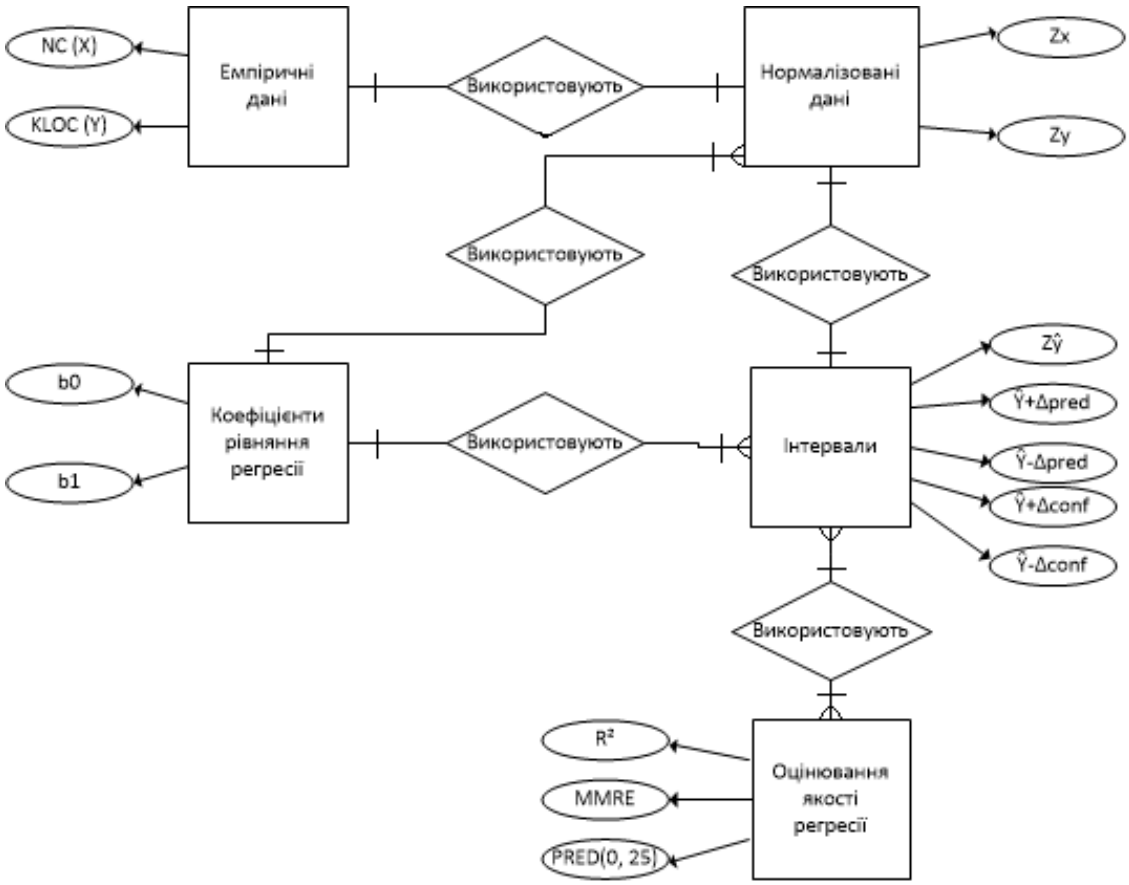

Рисунок 3.4 – Концептуальна модель програми

Отже, було розроблено концептуальну модель програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, у вигляді діаграми «Сутність-Зв’язок».

3.1.5 Розробка ескізу інтерфейсу користувача

Якість процесу інтерактивної взаємодії користувача із ПЗ пов’язана з такими психологічними характеристиками людини як пам’ять, час реакції та можливість сприйняття візуальної інформації. Тому, розроблюючи інтерфейс програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, потрібно врахувати і задовольнити такі основні властивості інтерфейсу, як:

- адаптованість;
- зручність користування;
- достатність;
- дружність;
- гнучкість.

З урахуванням всіх визначених вимог до властивостей інтерфейсу було спроектовано ескіз користувацького інтерфейсу головної форми програми, що розроблюється в даній кваліфікаційній роботі. Ескізи інтерфейсу програми представлено на рисунку 3.5.

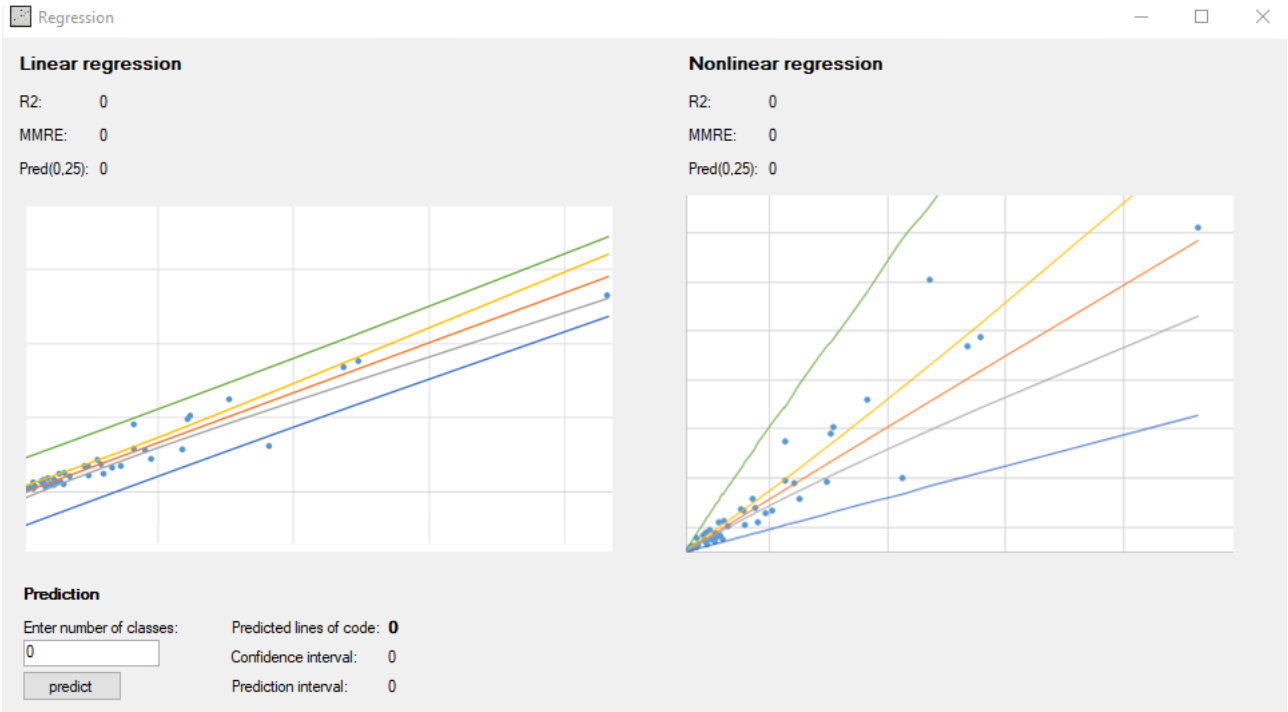

Рисунок 3.5 – Ескіз інтерфейсу програми

Отже, для програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, був розроблений інтерфейс користувача. Побудовані ескізи програми допоможуть задовольнити основні потреби користувачів.

3.2 Технічний проект програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд

На основі ескізного проекту було розроблено технічний проект програми. Для розробки програми «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд» на етапі технічного проекту було специфіковано класи програми, що розроблюється, а також побудовано статичну модель ПЗ, представлену діаграмою класів, та динамічну модель ПЗ, представлену діаграмою послідовності.

3.2.1 Побудова статичної моделі програми

Діаграма класів зображує набір класів, інтерфейсів та їх відношень. При моделюванні об’єктно-орієнтованих систем діаграми класів використовуються найчастіше. Діаграма класів є основною моделлю для статичного представлення системи [19].

На основі, побудованої в пункті 3.1.4 ескізного проекту, концептуальної моделі програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, необхідно побудувати статичну модель, у вигляді діаграми класів. Спроектована діаграма класів представлена на рисунку 3.6.

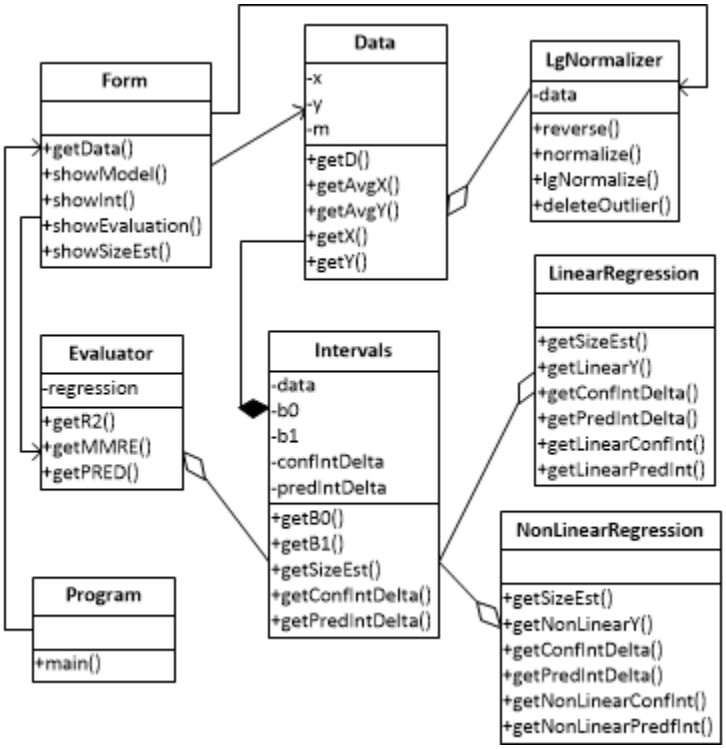

Рисунок 3.6 – Статична модель програми

Представлена на рисунку 3.6 діаграма класів є статичним представленням програми, що розроблюється в даній кваліфікаційній роботі.

3.2.2 Специфікації класів програми

Для класів, змодельованих у пункті 3.2.1 на статичній моделі програми, представленій діаграмою класів, необхідно розробити специфікації, що відображатимуть стани та поведінку програмного продукту.

1) Назва класу: «Program»

Короткий опис: клас «Program» відповідає за запуск програмного додатку.

Методи класу:

- main() – метод класу, що відповідає за запуск програмного додатку.

2) Назва класу: «Form»

Короткий опис: клас «Form» відповідає за відображення результатів розрахунків та побудови регресійних моделей, інтервалів передбачення, довірчих інтервлів, а також значень показників якості регресійних моделей на екран.

Методи класу:

- getData() – метод завантажує файл з вхідними емпіричними даними;

- showModel() – метод відображає побудовану регресійну модель на екран;

- showInt () – метод відображає довірчі інтервали та інтервали передбачення для побудованої регресійної моделі;

- showEvaluation() – метод відображає значення показників якості регресійної моделі: R^2, *MMRE, PRED*(0,25);

- showSizeEst() – метод відображає результати розрахунків для оцінювання розміру персонального застосунку.

3) Назва класу: «Data»

Короткий опис: клас «Data» містить вибірку випадкових величин, отриманих з файлу з вхідними емпіричними даними, відповідає за розрахунки значень вибіркової дисперсії, середнього вибіркового значень величин *X* та *Y*.

Атрибути класу:

- x – NC (кількість класів ПЗ);

- y – KLOC (кількість рядків коду ПЗ, у тисячах рядків коду);

- m – кількість підінтервалів розбиття вибірки випадкових величин.

Методи класу:

- getD() – метод здійснює розрахунок значення дисперсії вибірки випадкових величин;

- getAvgX() – метод здійснює розрахунок значення середнього вибіркового значень величин *X*;

- getAvgY() – метод здійснює розрахунок значення середнього вибіркового значень величин *Y*;

- getX() – метод повертає значення X;

- getY() – метод повертає значення Y.

4) Назва класу: «Intervals»

Короткий опис: клас «Intervals» відповідає за розрахунок значень коефіцієнтів регресійної моделі та побудову довірчих інтервалів і інтервалів передбачення.

Атрибути класу:

- b0 – перший коефіцієнт регресійної моделі;

- b1 – другий коефіцієнт регресійної моделі;

- data – вибірка випадкових величин;

- confIntDelta – границя довірчого інтервалу;

- predIntDelta – границя інтервалу передбачення.

Методи класу:

- getB0() – метод для отримання коефіцієнту регресії b_0;

- getB1() – метод для отримання коефіцієнту регресії b_1;

- getSizeEst() – метод повертає розрахункове значення оцінки розміру ПЗ;

- getConfIntDelta() – метод повертає границі довірчих інтервалів;

- getPredIntDelta() – метод повертає границі інтервалів передбачення.

5) Назва класу: «LgNormalizer»

Короткий опис: клас «LgNormalizer» відповідає за нормалізацію вхідних даних, перевірку нормальності закону розподілу даних, видалення викидів та застосування зворотнього до нормалізуючого перетворення.

Атрибути класу:

- data – вибірка випадкових величин.

Методи класу:

- reverse() – метод застосовує зворотне нормалізуюче перетворення.

- lgNormalize() – метод виконує нормалізацію вхідних даних на основі десяткового логарифму;

- Normalize() – метод перевіряє вибірку даних на нормальність закону розподілу;

- deleteOutlier() – метод відповідає за видалення викидів із вибірки даних;

6) Назва класу: «Evaluator»

Короткий опис: клас «Evaluator» відповідає за розрахунок показників якості регресійної моделі R^2, *MMRE, PRED*(0,25).

Атрибути класу:

- regression – екземпляр класу Intervals.

Методи класу:

- getR2() – метод розраховує і повертає значення показника якості R^2;

- getMMRE() – метод розраховує і повертає значення показника якості *MMRE*;

- getPRED() – метод розраховує і повертає значення показника якості *PRED* (0, 25).

7) Назва класу: «LinearRegression»

Короткий опис: клас «LinearRegression» відповідає за розрахунок границь довірчих інтервалів та інтервалів передбрчення для лінійної регресійної моделі та побудову лінійної регресійної моделі.

Методи класу:

- getSizeEst() – екземпляр класу Intervals, що повертає розрахункове значення *Y*;

- getLinearY() – повертає розрахункове значення *Y* за лінійним рівнянням регресії;

- getConfIntDelta() – екземпляр класу Intervals, що повертає границі довірчих інтервалів;

- getPredIntDelta() – екземпляр класу Intervals, що повертає границі інтервалів передбачення;

- getLinearConfInt() – повертає границі довірчих інтервалів для лінійної регресії;

- getLinearPredInt() – повертає границі інтервалів передбачення для лінійної регресії.

8) Назва класу: «NonLinearRegression»

Короткий опис: клас «NonLinearRegression» відповідає за розрахунок границь довірчих інтервалів та інтервалів передбрчення для нелінійної регресійної моделі та побудову нелінійної регресійної моделі.

Методи класу:

- getSizeEst() – екземпляр класу Intervals, що повертає розрахункове значення *Y*;

- getNonLinearY() – повертає розрахункове значення *Y* за нелінійним рівнянням регресії;

- getConfIntDelta() – екземпляр класу Intervals, що повертає границі довірчих інтервалів;

- getPredIntDelta() – екземпляр класу Intervals, що повертає границі інтервалів передбачення;

- getNonLinearConfInt() – повертає границі довірчих інтервалів для нелінійної регресії;

- getNonLinearPredInt() – повертає границі інтервалів передбачення для нелінійної регресії.

3.2.3 Динамічна модель програми

Динамічні моделі забезпечують представлення поведінки систем. «Динамізм» цих моделей полягає в тому, що в них відображається зміна станів у процесі роботи системи (в залежності від часу) [19].

На основі побудованої статичної моделі програми, представленої діаграмою класів, та розроблених діаграм діяльності для основних варіантів використання, необхідно побудувати динамічну модель програми. Спроектовану діаграму послідовності для програми оцінювання розміру

персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, представлено на рисунку 3.7.

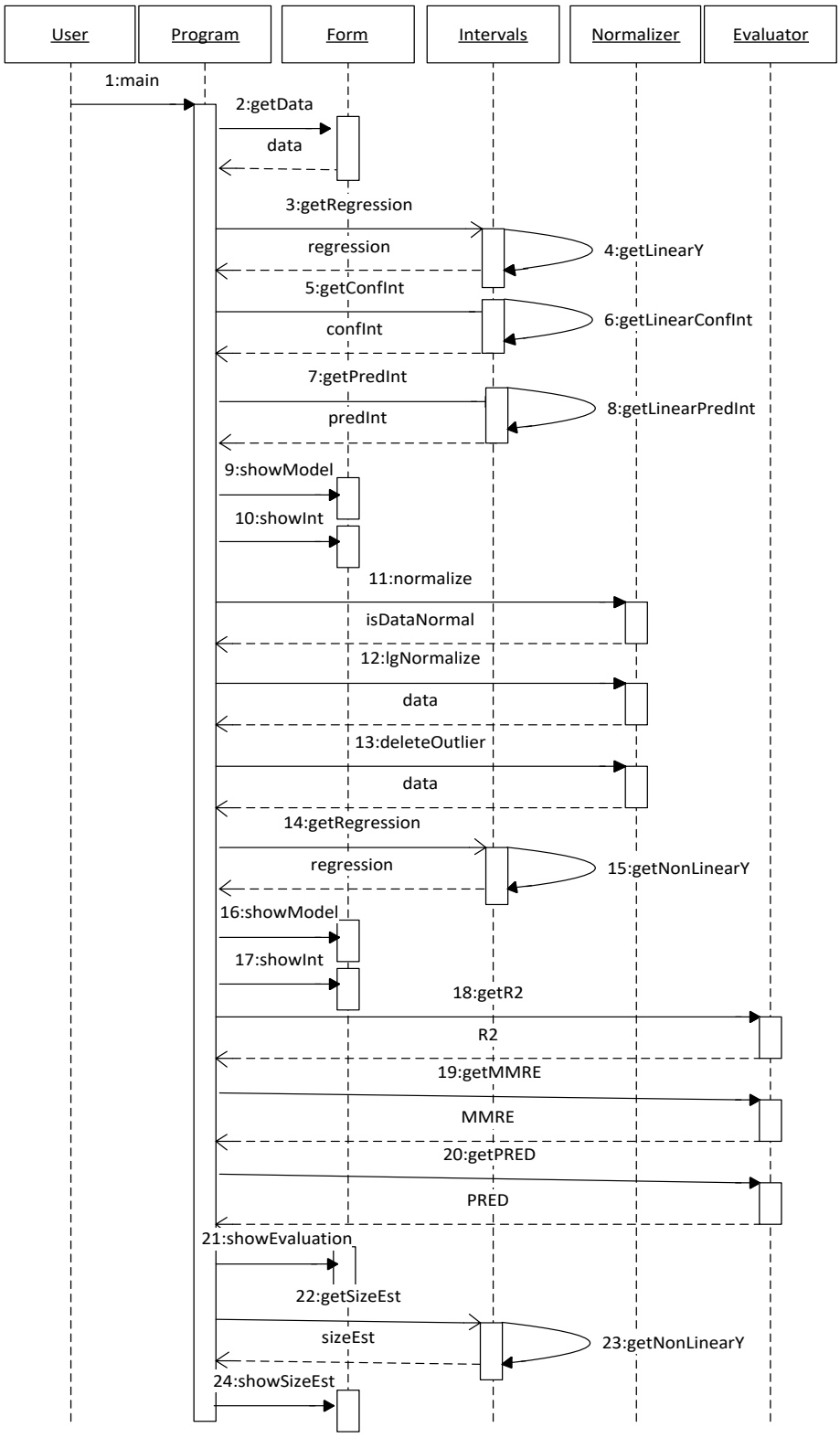

Рисунок 3.7 – Динамічна модель програми

Представлена на рисунку 3.7 діаграма послідовності є динамічним представленням програми, що розроблюється в даній кваліфікаційній роботі.

3.3 Робочий проект програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд

Для розробки програми «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд», на етапі робочого проекту було обрано мову програмування та середовище розробки. Також було розроблено діаграму компонентів ПЗ, здійснено кодування, тестування та випробування розробленого програми.

3.3.1 Обґрунтування вибору мови програмування та середовища розробки програмного забезпечення

Для розробки програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, було обрано мову програмування C#, середовище розробки Visual Studio та інтерфейс програмування додатків (API) Windows Forms.

C# – крос-платформна об'єктно-орієнтована мова програмування, розроблена корпорацією Microsoft. Мова C# близька до синтаксису C++ та Java, та спирається на практику застосування своїх попередників, серед яких, окрім раніше зазначених, також мови Object Pascal, Модула і Smalltalk. C# має строгу статичну типізацію і багато можливостей, що постійно розвиваються, включаючи безпеку типів, узагальнення, зіставлення шаблонів, асинхронізацію, записи та багато іншого. Як і Java, мова програмування C# підтримує можливість працювати з віртуальною машиною (середовище .NET), а також успадкувала концепці. байт-коду і більшої безпеки вихідного коду програм. На

відміну від C++, мова C# виключає деякі проблематичні моделі, пов'язані з безпечністю, такі як підтримка множинного успадкування класів [21].

Для розробки інтерфейсу користувача програми, що розроблюється в даній кваліфікаціцйній роботі, було обрано інтерфейс програмування додатків (API) Windows Forms, що є частиною .NET Framework. C# є найпопулярнішою мовою для розробки .NET. За допомогою .NET можна створювати додатки будь-якого типу, що працюють на будь-якій платформі. Від мобільних додатків, що працюють на iOS і Android, до корпоративних серверних додатків, що працюють на Windows Server і Linux, або великомасштабних мікросервісів, що працюють в хмарі [21].

Також, для розробки програми було обрано середовище розробки Visual Studio, оскільки воно надає змогу розробляти програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms. А також середовище розробки Visual Studio може використовуватись для всіх платформ, що підтримуються .NET Framework та Microsoft Windows, що є важливим для створення ПЗ, що розроблюється в даній кваліфікаційній роботі.

3.3.2 Діаграма компонентів програми

Діаграма компонентів моделюється для візуалізації основних компонентів ПЗ та взаємозв’язків між ними. Компонент — це фізичний контейнер для елементів моделі, таких як класи проектування чи файли з вихідними даними. [22]. Діаграму компонентів програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, наведено на рисунку 3.8.

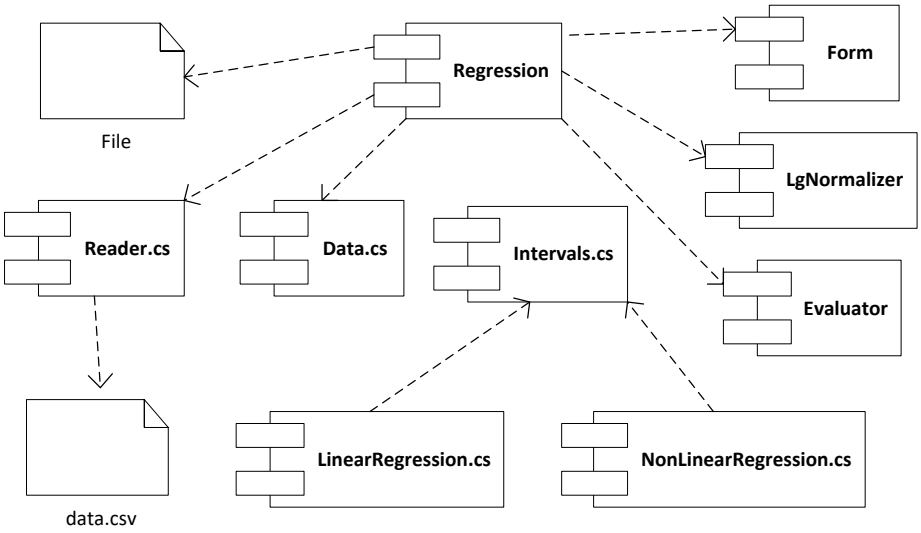

Рисунок 3.8 – Діаграма компонентів програми

Отже, на етапі розробки робочого проекту ПЗ була змодельовано діаграму компонентів для програми, що розроблюється в даній кваліфікаційній роботі.

3.3.3 Кодування та тестування програми

Реалізацію розробки програми для оцінювання розміру розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, було виконано мовою програмування C# в середовищі розробки Visual Studio.

Для тестування розробленого ПЗ застосуємо методи еквівалентного розбиття та «чорного ящика».

Класи еквівалентності для тестування функції «Введення вхідних даних» наведено в таблиці 3.11.

Таблиця 3.11 – Класи еквівалентності функції «Введення вхідних даних»

Вхідні дані	Правильні класи	Неправильні класи
Файл з даними розширення типу .csv	1. Обраний файл має розширення .csv та дані правильної структури	2. Файл має будь-яке роширення окрім .csv 3. Вхідні дані мають неправильну структуру

Тести з правильних класів еквівалентності наведено в таблиці 3.12.

Таблиця 3.12 – Тестування правильних класів еквівалентності функції «Введення вхідних даних»

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
1	1	Файл з розширенням типу .csv та даними правильної структури	Обраний файл завантажено успішно

Тести з неправильних класів еквівалентності наведено в таблиці 3.13.

Таблиця 3.13 – Тестування неправильних класів еквівалентності функції «Введення вхідних даних»

№ Тесту	№ покритого класу еквівалентності	Вхідні дані	Результат
1	2	Файл з розширенням типу .xlsx	Повідомлення про помилку у типі розширення файлу
2	3	Файл з даними пошкодженої структури	Повідомлення про помилку у структурі вхідних даних

Отже, із наведених таблиць можна зробити висновок, що тестування правильних та неправильних класів еквівалентності для функції «Введення вхідних даних» було пройдено успішно.

3.3.4 Випробування програми

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним нижче [23]:

1) Технічне заавдання.

2) Текст програми.

3) Опис програми.

4) Програма та методика випробувань.

5) Інструкція користувача.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище переліку [23].

Перевірка працездатності програми виконується згідно з вимогами до функціональних характеристик ПЗ. Перевірка вважається завершеною у разі відповідності складу і послідовності виконаних дій вимогам до функціональних характеристик.

Для програми для оцінювання розміру розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, було виконано випробування згідно з програмою та методикою випробувань, наведених у додатку Д – «Програма та методика випробувань ПЗ».

В процесі тестування було перевірено функціональність розробленого ПЗ, що полягало у введенні даних для тестування та проведенні аналізу отриманих результатів.

Випробування можливості введення вхідних даних у розроблене програмне забезпечення полягало у виконанні наступної послідовності дій:

- Запустити програму, натиснувши на іконку програмного додатку.

- Натиснути на пункт головного меню «File».

- Натиснути на опцію «Load».

- Обрати шлях до файлу з вхідними даними. розширення .csv.

В результаті файл з вхідними даними було успішно завантажено, а також розробленим ПЗ було здійснено необхідні розрахунки та виведено результати обчислень на екран користувача, що відповідало очікуваному результату випробування.

Випробування можливості оцінювання розміру персонального застосунку-органайзера, що створено мовою Kotlin для Андроїд, полягало у виконанні наступної послідовності дій:

- Ввести кількість класів у поле вводу «Enter number of classes».

- Натиснути на кнопку «predict».

В результаті виконання описаних вище дій було виконано оцінювання розміру персонального застосунку-органайзера, що створено мовою Kotlin для Андроїд, результати обчислень було виведено на екран користувача, що відповідало очікуваному результату випробування.

Отже, у результаті проведення випробувань розробленого ПЗ можна зробити висновок, що розроблене у даній кваліфікаційній роботі програмне забезпечення для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, повністю відповідає всім вимогам, визначеним у постановкі задачі до розробки ПЗ.

4 РЕЗУЛЬТАТ РОЗРОБКИ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПЕРСОНАЛЬНИХ ЗАСТОСУНКІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN ДЛЯ АНДРОЇД

В процесі виконання кваліфікаційної роботи, було побудовано нелінійну регресійну модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, із застосуванням нормалізуючого перетворення на основі десяткового логарифму.

На основі побудованої нелінійної регресійної моделі в ході виконання роботи була розроблена програма для оцінювання розміру персональних застосунків-органайзерів, що створюються для Android. Розроблена у кваліфікаційній роботі програма виконує наступні функції:

1) надання можливість вносити дані про проекти персональних застосунків (кількість рядків коду та кількість класів);

2) виконання нормалізації внесених даних за допомогою нормалізуючого перетворення на основі десяткового логарифму;

3) видалення викидів за наявності;

4) побудова лінійної регресійної моделі;

5) побудова довірчих інтервалів на основі нормалізованих даних;

6) побудова інтервалів передбачення на основі нормалізованих даних;

7) побудова нелінійної регресійної моделі на основі лінійного рівняння регресії;

8) розрахування метрик якості;

9) оцінювання розміру персональних застосунків.

Здійснено постановку задачі та розроблено технічне завдання для розробки програми «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд», що наведене в додатку А – «Технічне завдання».

У рамках даної кваліфікаційної роботи було виконано ескізний, технічний та робочий проекти.

На етапі ескізного проекту було виявлено основних акторів та варіанти використання ПЗ, та розроблено специфікації виявлених варіантів використання. Використовуючи мову моделювання UML було побудовано діаграму варіантів використання та концептуальну модель програми, також було розроблено діаграми діяльності для основних варіантів використання. Згідно зі специфікованими варіантами використання та побудованими сценаріями, представленими у вигляді діаграм діяльністі, було розроблено ескіз користувацького інтерфейсу.

На етапі технічного проекту було специфіковано класи програми, що розроблюється, а також побудовано статичну модель ПЗ, представлену діаграмою класів, та динамічну модель ПЗ, представлену діаграмою послідовності.

На етапі робочого проекту було обрано мову програмування та середовище розробки. Також було розроблено діаграму компонентів ПЗ, здійснено кодування, тестування та випробування розробленого програми.

Реалізацію розробки програми було виконано мовою програмування C# в середовищі розробки Visual Studio.

Тестування програми показало, що розроблене ПЗ для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Android, успішно виконує всі поставлені перед ним задачі. Текст програми представлений у додатку Б – «Текст програми».

Випробування програми показало, що розроблене ПЗ повністю відповідає усім вимогам, сформульованим у технічному завданні, що наведене у додатку А. Програма та методика випробувань наведені у додатку Д – «Програма та методика випробувань ПЗ».

Опис програми наведено у додатку В – «Опис програми».

Для експлуатації програми також було розроблено інструкцію користувача, котру наведено в додатку Г – «Інструкція користувача».

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІД РОЗРОБКИ ТА ВПРОВАДЖЕННЯ ПРОГРАМИ «НЕЛІНІЙНА РЕГРЕСІЙНА МОДЕЛЬ ДЛЯ ОЦІНЮВАННЯ РОЗМІРУ ПЕРСОНАЛЬНИХ ЗАСТОСУНКІВ-ОРГАНАЙЗЕРІВ, ЩО СТВОРЮЮТЬСЯ МОВОЮ KOTLIN ДЛЯ АНДРОЇД»

5.1 Вступ

Дана кваліфікаційна робота присвячена розробці програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Android.

Економічна ефективність – це - результативність економічної системи, виражена у співвідношенні корисних кінцевих результатів її функціонування, у даному випадку розробленого програми, до витрачених ресурсів [24]. Ефективність інформаційних систем визначається ступенем їх позитивного впливу на підприємство або процес в результаті використання засобів обчислювальної техніки, ПЗ, зміни інформаційних потоків та організаційної структури управління.

Створення ПЗ вимагає одноразових витрат на розробку, придбання необхідних технічних засобів і поточних витрат на функціонування продукту. Економія від впровадження ПЗ визначається з урахуванням витрат на його експлуатацію. Економічні показники визначаються за діючими на момент розрахунку оптовими цінами, тарифами, ставками заробітної плати та податків.

Для обґрунтування доцільності розробки програми 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 у вигляді річного економічного ефекту, а також строк окупності розробленого програми.

5.2 Розрахунок витрат на створення та експлуатацію ПЗ

Для розрахунку витрат на розробку програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, необхідно обчислити витрати: на оплату праці розробника, на експлуатаційні витрати для комп`ютера, на якому відбувається розробка, на засоби розробки та витрат на виробничі запаси.

Розробка ПЗ виконується програмістом, місячний оклад якого складає 10 000 грн. Додаткова заробітна плата складає 20% від основної – 2 000 грн. Отже, основна і додаткова заробітна плата складає 12 000,00 грн/міс. Вартість сучасного комп`ютера складає 16 000 грн. При вартості кіловат-години електроенергії 1,68 грн., розраховується вартість розробки програми. Витрати на допоміжні матеріали приведені в табл. 5.1.

Таблиця 5.1 - Витрати на допоміжні матеріали

Найменування матеріалу	Вартість, грн.
Офісний папір (150 аркушів)	80
Заправка картриджу тонером для принтера	100
Доступ до мережі Internet	200
Флеш-накопичувач даних	250
Сума	530

Вартість розробки ПЗ розрахуємо за формулою:

$$Впз = (Взп + Вс + Вз + Ве) * Т + Вдм \quad (5.1)$$

де: *Взп* – основна і додаткова заробітна плата розробника, грн.;

Вс – витрати на відрахування в соціальні фонди (38% від основної і додаткової заробітної плати), грн.;

Вз – загально-виробничі витрати (10% від основної заробітної плати), грн.;

Ве – витрати на електроенергію;

Т – тривалість розробки, міс;

Вдм – витрати на допоміжні матеріали.

Витрати на електроенергію за умови споживання 0,5 кВт, тривалості роботи за місяць 21*8=168 годин, що у підсумку складає:

$$Ве=168*1,68*0,5=141,12 \text{ грн.}$$

Таблиця 5.2 – Витрати на розробку програми

Найменування	Сума (грн.)
Заробітна плата розробника	12 000
Витрати на відрахування в соціальні фонди	4 560
Загально-виробничі витрати	1 200
Витрати на електроенергію	141,12
Витрати на допоміжні матеріали	530

Відповідно до формули (5.1) вартість розробленого ПЗ складає:

$$Впз = (12\,000 + 4\,560 + 1\,200 + 141,12) * 2 + 530 = 36\,332,24 \text{ грн.}$$

Експлуатаційні витрати для комп'ютера розрахуємо за формулою:

$$Вк = А + Вм + Е \quad (5.2)$$

де: *А* – амортизаційні відрахування, грн.

Вм – річні витрати на основні і допоміжні матеріали, грн.

Е – витрати на електроенергію, грн.

Амортизаційні відрахування на основні засоби (комп'ютер) складають 25% від первісної вартості в рік: $А = 16\,000 * 0,25 = 4\,000$ грн.

У масштабах підприємства річні витрати на основні і допоміжні матеріали визначаються в розмірі 5% вартості основних засобів: *Вм* = 16 000* 0,05 = 800 грн.

Річний обсяг роботи комп`ютера у годинах визначається за формулою:

$$P = 253,3 * Тз,$$

де *Тз*- середнє денне завантаження устаткування (близько 7 годин).

253,3 – середня кількість робочих днів у році.

Отже, річний обсяг роботи комп`ютера складе: *Р* = 253,3 * 7 = 1 773,1 години.

Витрати на електроенергію при 1 773,1 годинах роботи устаткування в рік складуть: *Е* = 1 773,1* 1,68 * 0,5 = 1 489,41 грн.

За формулою (5.2), експлуатаційні витрати для комп`ютера за рік складуть:

$$Вк = 4 000 + 800 + 1 489,41 = 6 289,41 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію ПЗ складуть:

$$В = 36 332,24 + 6 289,41 = 42 621,65 \text{ грн.}$$

5.3 Економічна ефективність розробки та впровадження програми

Основним показником економічної ефективності функціонування розробленого ПЗ є зменшення витрат, підвищення ефективності роботи та зниження трудомісткості розрахункових робіт щодо оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд. Розроблене у кваліфікаційній роботі ПЗ дозволить підвищити якість та швидкість керування інформацією, а також знизить ризики допущення помилок, що є вагомим показником економічної ефективності. Впровадження розробленого програми дозволить частково вивільнити робочий час працівників та значно підвищити продуктивність праці.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що впровадження програмного продукту вивільняє 0,5 працівника за експертною оцінкою фахівців підприємства.

Заробітна платня 0,5 працівника в рік складає: 10000*12*0,5=60 000,00 грн. Економічний ефект першого року розраховується за формулою:

$$Ерік = \Delta Сп - Ен*К \quad (5.3)$$

де: $\Delta Сп$ – вивільнені кошти після впровадження ПЗ мінус експлуатаційні витрати; $\Delta Сп$ = 60 000 – 12 214,41 = 47 785,59 грн.

Ен – нормативний коефіцієнт економічної ефективності (дорівнює коефіцієнту амортизації 0,25%);

К – капітальні витрати на впровадження ПЗ (36 332,24 грн).

За формулою (5.3), економічний ефект першого року дорівнює:

$$Ерік = 47\ 785{,}59 – 0{,}25*36\ 332{,}24 = 38702{,}53 \text{ грн.}$$

Строк окупності капітальних вкладень визначається за формулою:

$$Т = \frac{К}{\Delta Сп} = 0{,}76 \text{ року.}$$

Отже строк окупності програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, складає приблизно 9 місяців.

Виходячи з розрахунків, можна зробити висновок, що розробка програми для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, є економічно вигідною.

6 ОХОРОНА ПРАЦІ

6.1 Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [25].

Мета охорони праці – забезпечення безпечних, нешкідливих і сприятливих умов праці через вирішення багатьох складних завдань, основними з яких є:

– проектування підприємств, технологічних процесів і конструювання обладнання з обов'язковим виконанням вимог охорони праці;

– знаходження оптимальних співвідношень між різними факторами виробничого середовища, що дозволяє забезпечити мінімум несприятливого впливу їх на здоров'я працівників;

– встановлення, законодавче оформлення визначених норм кожного з несприятливих або небезпечних факторів, систематичний король за їх застосуванням;

– розробка конкретних заходів щодо покращення умов праці та забезпечення її безпеки на основі застосування у виробництві новітніх досягнень науки та техніки;

– застосування раціональних засобів захисту працівників від впливу несприятливих факторів виробничого середовища, а також втілення організаційних заходів, які нейтралізують або послаблюють ступінь їх впливу на організм людини;

– розробка та застосування методів і засобів оцінки ефективності заходів з охорони праці, що плануються і здійснюються.

Основним законом в галузі охорони праці є Закон України «Про охорону праці». Дія Закону України «Про охорону праці» поширюється на всіх юридичних та фізичних осіб, які відповідно до законодавства використовують найману працю, та на всіх працюючих [25].

6.2 Аналіз шкідливих та небезпечних факторів на робочому місці

Під час укладання трудових договорів роботодавець повинен поінформувати працівника під розписку про умови праці та про наявність на його робочому місці небезпечних і шкідливих виробничих факторів, які ще не усунуто, можливі наслідки їх впливу на здоров’я та про права працівника на пільги і компенсації за роботу в таких умовах відповідно до законодавства і колективного договору [25].

Небезпечний виробничий фактор - фактор середовища і трудового процесу, що може бути причиною гострого захворювання (отруєння), раптового різкого погіршення здоров’я або смерті [26].

Шкідливий виробничий фактор - фактор середовища або трудового процесу, вплив якого на працівника за певних умов (інтенсивність, тривалість дії тощо) може спричинити професійне або виробничо обумовлене захворювання, тимчасове або стійке зниження працездатності, підвищення частоти соматичних та інфекційних захворювань, призвести до порушення здоров’я як працівника, так і його нащадків [26].

В таблиці 6.1 перераховані шкідливі та небезпечні виробничі фактори, з якими можуть зіткнутися під час роботи розробники.

Таблиця 6.1 – Шкідливі та небезпечні виробничі фактори

Найменування фактору	Можливі джерела його виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що загоряються і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Електромагнітне випромінювання в тому числі і рентгенівське	ЕПТ (Дисплей є джерелом рентгенівського, радіочастотного, ультрафіолетового й інфрачервоного випромінювання та випромінювання звукового діапазону)	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

Тривала і інтенсивна робота на комп'ютері може стати джерелом важких професійних та звичайних захворювань таких як:

– неправильна постава - це призводить до подальшого розвитку викривлення хребта сколіозу, лордозу, кіфозу, а як результат головні болі, болі в області шиї і всього хребта, болі в області таза. Неправильно положення ніг може привести до артриту (запалення суглобів), артрозу (деформації);

– тунельний синдром - найвідоміше захворювання людей працюють за комп'ютером. Воно ж синдром зап'ястного каналу. Тунельний синдром проявляється після кількох годин напруженої роботи за комп'ютером;

– геморой - хвороба аналогічна варикозного розширення вен. Причиною хвороби може бути застій крові у венах;

– розвиток короткозорості - через те, що екран монітора за контрастом вище, ніж навколишні об'єкти розвивається короткозорість;

– порушення фокусування - наслідком напруженої роботи за монітором є порушення фокусування, яка може бути викликана перенапруженням очних м'язів;

– сухість очей - через рефлекс, заснованого на тому, що при погляді на джерело світла очей починає менше моргати виникає «обсушування» рогівки ока яке призводить до очних болів.

6.3 Розрахунок системи штучного освітлення у офісному приміщені з екранними пристроями

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. До сучасного виробничого освітлення висуваються наступні вимоги:

– відповідність рівня освітленості робочих місць характеру роботи, що здійснюється;

– досить рівномірний розподіл яскравості на робочих місцях поверхнях і у навколишньому просторі;

– неперервність освітленості у часі;

– оптимальна направленість світлового потоку, який випромінюють світлові прилади;

– довговічність, економічність, електро- і вогненебезпека, естетичність, зручність і простота експлуатації.

Штучне освітлення за конструктивним виконанням може бути загальним і місцевим. При загальному освітленні всі робочі місця отримують освітлення від загальної освітлюваної установки. Комбіноване освітлення поряд із загальним включає місцеве освітлення, що зосереджує світловий потік безпосередньо на робочих місцях. Використання лише місцевого освітлення не допустимо, оскільки виникає необхідність постійної адаптації зору, створюються тіні та інші негативні фактори.

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.

Для розрахунку системи штучного освітлення офісного приміщення завдовжки 5 м і завширшки 6 м використаємо метод на основі коефіцієнта світлового потоку, що вимірюється в люменах (Лм).

Визначення світлового потоку (*F*), що падає на поверхню, розраховується за формулою:

$$F = \frac{Emin * k_з * S * z}{\eta} \text{ (Лм)}, \quad (6.1)$$

де: *Emin* – нормована мінімальна освітленість, Лк (визначається за таблицею). В нашому випадку *Emin* =300Лк;

$k_з$ – коефіцієнт запасу, який враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації. В нашому випадку $k_з$ =1,5;

S – площина приміщення, яке освітлюється (*S*=30м2);

z – коефіцієнт використання світлового потоку з урахуванням коефіцієнту відображення стелі =70% і стін =50%; виражається відношенням світлового потоку, який падає на розрахункову поверхню, до сумарного потоку всіх ламп і виражається у долях одиниці.

Значення η визначається за таблицею коефіцієнтів використання. Для цього розрахуємо індекс приміщення (*i*) за формулою:

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

де: *h* – розрахункова висота підвісу, *h*=2,7м;

A – ширина приміщення, *A*=5м;

B – довжина приміщення, *B*=6м.

Отже, індекс приміщення *i* = 1.01.

За таблицею знайдемо коефіцієнт використання люмінесцентних світильників η = 0,32. Підставимо отирмані значення у формулу (6.1), отримаємо: *F*= 49 500 Лм.

Для освітлення приміщення даного відділу обираємо люмінесцентні лампи типу ЛДУ 80, світловий потік яких дорівнює 4070Лм. Розрахуємо необхідну кількість світильників для приміщення з комп'ютерами при загальному рівномірному освітленні.

Кількість необхідних ламп N розраховується за формулою:

$$N = \frac{F}{Fn},$$

де Fn - світловий потік, 4070Лм.

Отримаємо: N = 13,6 ≈ 14 шт. Із розрахунку освітленості виробничого приміщення виходить, що для правильної подачі світла і правильної освітленості робочих місць відділу необхідно використовувати світильники типу УПМ. Кожний світильник комплектувати 2 лампами. Розмістити світильники двома паралельними рядами по три в кожному ряді і один світильник перпендикулярно двом паралельним у середині приміщення.

6.4 Заходи щодо зменшення впливу шкідливих факторів

Працюючи за комп'ютером, рекомендуємо дотримуватися правил тривалості роботи, правильної постави, розміру шрифтів та зображень, вимог до приміщення тощо.

Принципи правильної роботи за комп'ютером:

– у робочому приміщенні (кімнаті), де встановлені комп'ютери, щодня потрібно виконувати вологе прибирання;

– приміщення, у якому знаходяться комп’ютери, потрібно провітрювати щогодини;

– після кожного часу роботи рекомендується робити десяти хвилинну перерву, яку зручно суміщати з провітрюванням. За будь-яких умов безперервна робота за комп’ютером для дорослої людини не повинна перевищувати двох годин. Під час перерви не варто читати або дивитися телевізор;

– необхідно постійно слідкувати за станом екрану монітора: він має бути чистим, без плям та пилу;

– потрібно слідкувати за поставою: ноги повинні твердо стояти на підлозі чи на спеціальній підставці; стегна повинні бути розташовані під прямим кутом до тулуба, а гомілки – під прямим кутом до стегон; сидіти потрібно прямо або злегка нахилившись вперед; відстань від очей до екрана монітора – не менше 55-60 см; центр екрану має знаходитися на рівні очей чи трохи нижче; рекомендується хоча б раз на день виконувати гімнастику для очей;

– щоб попередити „синдром сухого ока”, потрібно моргати кожні 3-5 секунд;

– не використовувати звичайний телевізор замість монітора;

– у процесі роботи рекомендується періодично (приблизно раз на 20-30 хвилин) переводити погляд з екрана на найбільш віддалений предмет у кімнаті, а ще краще – на віддалений об’єкт за вікном;

– якщо з’явилося відчуття втоми, напруження, сонливості, тяжкості в очах, потрібно припинити роботу та хоча б трохи відпочити.

Рекомендовані умови для роботи за комп'ютером:

– твердий стілець. край стільця не повинен тиснути на судини під колінами;

– відстань до монітора повинна бути 50-70 см;

– перерву в сидячій роботі, 15-20 хвилин кожні 1-2 години;

– правильне освітлення робочого місця. при слабкому світлі очі напружуються і болять;

– потрібно закривати очі для відпочинку. час від часу відводите очі вбік, щоб дати відпочити своєму зору.

7 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

7.1 Забруднення навколишнього середовища в процесі використання комп’ютера

Під навколишнім середовищем розуміють цілісну систему взаємопов’язаних природних і антропогенних об’єктів і явищ, під впливом і при безпосередньому використанні яких відбувається праця, побутова діяльність, відпочинок людей. Поняття «навколишнє середовище» включає соціальні, природні і штучно створені фізичні, хімічні та біологічні фактори, тобто все те, що впливає на життя і діяльність людини.

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.

Економічна діяльність у всіх її проявах здійснює забруднення навколишнього середовища. Нині рівень забруднення досяг загрозливих розмірів, набувши по суті кризового характеру.

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, переселених 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.

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.

Отже, 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

Для вирішення питання забруднення навколишнього середовища у процесі експлуатації комп'ютерів виробниками комп'ютерних технологій 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.

Для того, щоб зменшити рівень забруднення навколишнього середовища, варто вжити певних заходів, які мінімізують цей вплив, якщо його неможливо усунути цілком. 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.

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.

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.

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.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було удосконалено однофакторну нелінійну регресійну модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, на основі нормалізації за допомогою десяткового логарифма та викидів регресії, та було розроблено програму для оцінювання розміру обраного ПЗ. Ця нелінійна регресійна модель у порівнянні з існуючою моделлю має більшу достовірність оцінювання по параметрам якості *MMRE* і *PRED*(0,25) та вужчі довірчий інтервал і інтервал передбачення. Поставлені завдання було виконано у повному обсязі, а саме:

- проведено аналіз сучасних методів та моделей для оцінювання розміру персональних застосунків-оганайзерів, що створюються мовою Kotlin для Андроїд;

- здійснено обґрунтування актуальності побудови регресійної моделі для оцінювання розміру персональних застосунків-оганайзерів, що створюються мовою Kotlin для Андроїд;

- зібрано вихідні дані з завершених проектів персональних застосунків-оганайзерів, що створюються мовою Kotlin для Андроїд, які були використані для побудови удосконаленої регресійної моделі;

- здійснено побудову регресійної моделі шляхом переходу від лінійної регресійної моделі до нелінійної, із застосуванням зворотного нормалізуючого перетворення на основі десяткового логарифму.

В даній кваліфікаційній роботі було проведено огляд літературних джерел, нормативних документів та Internet-джерел для пошуку та аналізу чинних програмних рішень, методів і моделей для оцінювання розміру ПЗ.

За допомогою побудованої нелінійної регресійної моделі з використанням нормалізуючого перетворення на основі десяткового логарифму було

підвищено достовірність оцінювання розміру персональних застосунків-органайзерів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Управление проектами: Менеджмент ИТ-проекта. URL: https://habr.com/ru/post/169693/ (дата звернення: 20.09.22).

2. Котов С.Л. Информационно-аналитическая система оценивания трудозатрат и стоимости создания программных средств. *Программные продукты и системы.* Том 30, № 3. 2017. С. 469-474.

3. Метрики кода и практическая реализация по их сбору и анализу. Часть 1. Метрики. URL: https://club.cnews.ru/blogs/entry/metriki_koda_i_prakticheskaya _realizatsiya_po_i h_sboru_i_analizu_chast_1_ (дата звернення: 26.09.22).

4. Телехан А.М., Макарова Л.М. Нелінійна регресійна модель для оцінювання розміру персональних застосунків органайзерів, що створюються мовою Kotlin для Андроїд. *Інформаційні технології: моделі, алгоритми, системи (ITMAS-2022): Матеріали III Всеукраїнської науково-практичної інтернет-конференції* (26-28 жовтня 2022 р.). Миколаїв: НУК імені адмірала Макарова, 2022. С.27-28. Режим доступу: http://itconf.nuos.edu.ua/2022/proceedings/

5. Введение в язык Kotlin. URL: https://metanit.com/kotlin/tutorial/1.1.php (дата звернення: 28.09.22)

6. Kotlin is now Google’s preferred language for Android app development. URL: https://techcrunch.com/2019/05/07/kotlin-isnow-googles-preferred-language-for-android-app-development/ (дата звернення: 28.09.22).

7. The Kotlin Programming Language. URL: https://github.com/JetBrains/kotlin/ (дата звернення: 28.09.22).

8. Android. Проект с открытым исходным кодом. URL: https://source.android.com (дата звернення: 03.10.22).

9. Табунщик Г.В., Кудерметов Р.К., Брагіна Т.І. Інженерія якості програмного забезпечення: навчальний посібник. Запоріжжя: ЗНТУ. 2013. 180 с.

10. Титов А.И. Выбор метрики размера проекта в модели оценки трудоемкости разработки программ. *Интеллектуальные технологии на транспорте.* 2016. С.31-38.

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

12. Тютюнников Н.Н. Оценка размера создаваемого программного средства с использованием функциональных точек. Перспективы развития информационных технологий. №18. 2014. 207 с.

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

14. Prykhodko S., Prykhodko N., Makarova L., Pukhalevych A. Outlier Detection in Non-Linear Regression Analysis Based on the Normalizing Transformations. *Proceedings of the 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), IEEE.* (Lviv-Slavske, 2020). P. 407-410.

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

16. Приходько С.Б., Макарова Л.М., Пугаченко К.С. Методичні вказівки та завдання до виконання лабораторних робіт з дисципліни "Обробка експериментальних даних на комп`ютері". Миколаїв: НУК, 2018. 76 с.

17. Макарова Л.М., Приходько Н.В., Кудін О.О. Побудова нелінійної регресійної моделі для оцінювання розміру веб-додатків, реалізованих мовою

Java. *Вісник Херсонського національного технічного університету.* Херсон, 2019. № 2(69). С. 145-153.

18. Буч Г., Рамбо Д., Якобсон И. Краткая история UML, язык UML. Руководство пользователя. 2-е издание. СПб.: Питер – Принт. 2006. 496 с.

19. Орлов С.А. Технологии разработки программного обеспечения. СПб.: Питер. 2002. 464 с.

20. Кузнецов С.Д. Основы баз данных. М:Бином. 2-е издание. 2007. 488 с.

21. C# - the modern, innovative, open-source programming language for building all your apps. URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата звернення: 12.10.22).

22. The Unified Modeling Language for Object-Oriented Development. Documentation set. ver. 1.1. Rational Software Corp. September 1997.

23. Методика проведення перевірки комплектності програмної документації. URL: https://ukrbukva.net/page,11,108340-Baza-dannyh-tehnologicheskogo-oborudovaniya-dlya-proizvodstva-pechatnyh-plat.html (дата звернення: 29.10.2022).

24. Енциклопедія сучасної України. URL: https://esu.com.ua/ (дата звернення: 01.11.2022).

25. Про охорону праці : Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення:04.11.2022).

26. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу : Державні санітарні норми та правила від 08.04.2014 р. №248. Дата оновлення: 08.04.2014. URL: https://zakon.rada.gov.ua/laws/show/z0472-14#n14. (дата звернення:04.11.2022).

Додаток А - Технічне завдання

ВСТУП

Для полегшення процесу планування проектів з розробки персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, необхідно створити програму, котра дозволить отримати прогнозований розмір персонального застосунку-органайзеру на основі побудованої нелінійної регресійної моделі.

Найменування програми, що розробляється – «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд».

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Розробка програми виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Експлуатаційне призначення

Експлуатаційним призначенням програми є використання ПЗ розробниками для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд.

2.2 Функціональне призначення

Функціональним призначенням програми є побудова нелінійної регресійної моделі для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд, оцінювання якості побудованої моделі, передбаченя розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд та виведення розрахунків на екран користувача.

3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати всі нижчеописані функції.

– надання можливість вносити дані про проекти персональних застосунків-органайзерів (кількість рядків коду та кількість класів);

– виконання нормалізації внесених даних за допомогою нормалізуючого перетворення на основі десяткового логарифму;

– видалення викидів за наявності;

– побудова лінійної регресійної моделі;

– побудова довірчих інтервалів на основі нормалізованих даних;

– побудова інтервалів передбачення на основі нормалізованих даних;

– побудова нелінійної регресійної моделі на основі лінійного рівняння регресії;

– розрахування метрик якості;

– оцінювання розміру персональних застосунків-органайзерів;

– виведення отриманих розрахунків на екран користувача.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення даних (клавіатура та миша). Дані мають бути організовані у вигляді файлу з розширенням типу .csv та вводитись у систему за допомогою вибору шляху до відповідного файлу у вікні операційної системи.

Вихідні дані: виведення даних здійснюється за допомогою пристроїв виведення даних (монітор користувача).

3.2 Вимоги до надійності

Програмне забезпечення повинно функціонувати без збоїв при умові безперебійної роботи комп’ютера. При виникненні збоїв в роботі комп’ютера, відновлення нормальної роботи програмного забезпечення повинне проводитися після перезавантаження програмного продукту.

У випадку втрачення частини інформації програмне забезпечення повинно продовжувати коректне функціонування. У випадку неможливого продовження коректної роботи має бути виведене відповідне повідомлення користувача ПЗ.

У випадку введення користувачем програмного забезпечення некоректної інформації – обрання некоретного щляху до файлу з даними, чи файлу з даними некоректного формату, ПЗ повинно вивести відповідне повідомлення про помилку користувача та надати користувачу можливість виправити відповідну помилку.

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 та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 500 MB вільного місця;

– наявність пристроїв вводу та виводу: монітор, клавіатура, миша

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування програмного забезпечення необхідно використовувати ОС Windows версії не нижче 7.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна містити:

– технічне завдання (ТЗ);

– текст програми;

– опис програми;

– інструкцію користувача;

– програму і методику випробувань.

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 представлені в таблиці А.1.

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадія розробки	Етапи робіт	Вміст робіт	Контрольні точки
1	2	3	4
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів, обґрунтування необхідності проведення досліджень,	Початок: 01.09.22 Закінчення 13.09.22
	Розробка та затвердження ТЗ.	Визначення вимог до ПЗ. Узгодження і затвердження ТЗ.	
Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних. Уточнення методів рішення задачі. Розробка загального опису алгоритму розв'язання задачі	Початок: 14.09.22 Закінчення 28.09.22
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проекту	
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних. Розробка алгоритму рішення задачі. Розробка структури ПЗ	Початок: 29.09.22 Закінчення 13.10.22
	Затвердження технічного проекту	Розробка пояснювальної записки. Узгодження і затвердження технічного проекту	
Робочий проект	Розробка ПЗ	Програмування та налагодження програми	Початок: 14.10.22 Закінчення 03.11.22
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	
	Випробування ПЗ	Розробка, узгодження та затвердження порядку і методики випробувань. Проведення попередніх випробувань. Коригування ПЗ і програмної документації за результатами випробувань.	

Приймання проводиться відповідно до програми і методики випробувань і документується за допомогою протоколу проведення випробувань. У разі

знаходження помилок під час приймання програмного забезпечення, складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджуються керівником організації–замовника та організації–розробника. Розробник повинен, на протязі не більше ніж 2 тижні, виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б - Текст 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]);
                                }
                                stream.Close();
                        }
                }
                if (reader != null)
                {

                double[][] inputData = reader.get_Data();

                LinearRegression linearRegression = new LinearRegression(new
        Data(inputData));

                Intervals intervals1 = new Intervals(linearRegression, chartLinear);

                R2_lin_value.Text = getFormattedR2Evaluation(linearRegression);
                mmre_lin_value.Text =
        getFormattedMMREvaluation(linearRegression);
                PRED25_lin_value.Text =
        getFormattedPREDEvaluation(linearRegression);

                Data normalData =
        LgNormalizer.lgNormalizeAndDeleteOutliers(inputData);
                nonlinearRegression = new NonlinearRegression(new
        LinearRegression(normalData));

                Intervals intervals2 = new Intervals(nonlinearRegression,
        chartNonlinear);

                r2_nonlin_value.Text =
        getFormattedR2Evaluation(nonlinearRegression);
                mmre_nonlin_value.Text =
        getFormattedMMREvaluation(nonlinearRegression);
                PRED25_nonlin_value.Text =
        getFormattedPREDEvaluation(nonlinearRegression);
                classes_numeric.Enabled = true;
                predict_button.Enabled = true;
                }
                else
                {
                        MessageBox.Show("Error! Please load .csv file");
                        return;
                }
        }
        catch (Exception err)
        {
        MessageBox.Show("Selected file has incorrect data structure or wrong
extension");
        clearPrediction();
```

```
                MessageBox.Show(err.Message);
                }
        }
}

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

private string getFormattedMMREvaluation(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)
{
        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 = getFormattedPredIntlDelta(classes);
        confidence_delta_label.Text = getFormattedConfIntDelta(classes);
}

private string getFormattedPredIntDelta(int classes)
{
        return getFormattedIntDelta(nonlinearRegresion.getPredIntDelta(classes));
}

private string getFormattedConfIntDelta(int classes)
{
        return getFormattedIntDelta(nonlinearRegression.getConfIntDelta(classes));
}

private string getFormattedIntDelta(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;
		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++;
		}

		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();
            }

            public int get_nLines()
            {
                return nLines;
            }

            public int get_nColumns()
            {
                return nColumns;
            }

            public double[][] get_Data() => data;
            public string[] get_Header()
            {
                return header;
            }
        }
}
```

lgNormalizer.cs

```
using System;
namespace Regression
{
        static class lgNormalizer
        {
                public static double lgNormalize(double numb)
                {
                        return Math.Log(numb);
                }
                public static double reverse(double numb)
                {
                        return Math.Round(Math.Exp(numb));
                }
                public static Data lgNormalizeDeleteOutliers(double[][] inputData)
                {
                        double[][] nData = lgNormalizeData(inputData);
                        while (hasOutliers(nData))
                        {
                                nData = deleteOutliers(nData);
```

```
        }
        return new Data(nData);
}
public static double[][] lgNormalizeData(double[][] data)
{
        double[][] lgNormalizedData = new double[data.Length][];
        for (int i = 0; i < data.Length; i++)
        {
                lgNormalizedData[i] = new double[2];
                lgNormalizedData[i][0] = lgNormalize(data[i][0]);
                lgNormalizedData[i][1] = lgNormalize(data[i][1]);
        }
        return lgNormalizedData;
}

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 hasOutliers(double[][] data)
{
        bool flag = false;
        LinearRegression regr = new LinearRegression(new Data(data));
        double[][] clearData = new double[data.Length][];
        double[][] posPredInt = regr.getPosPredInt();
        double[][] negPredInt = regr.getNegPredInt();
        for (int i = 0; i < data.Length; i++)
        {
                if (data[i][1] > posPredInt[i][1] || data[i][1] < negPredInt[i][1])
                {
                        flag = true;
                        break;
                }
        }
        return flag;
}

private static int countOutliers(double[][] data)
{
        int cnt = 0;
        LinearRegression regr = new LinearRegression(new Data(data));
```

```
                double[][] clearData = new double[data.Length][];
                double[][] posPredInt = regr.getPosPredInt();
                double[][] negPredInt = regr.getNegPredInt();
                for (int i = 0; i < data.Length; i++)
                {
                        if (data[i][1] > posPredInt[i][1] || data[i][1] < negPredInt[i][1])
                        {
                                cnt++;
                        }
                }
                return cnt;
        }

        public static double[][] deleteOutliers(double[][] data)
        {
                LinearRegression regr = new LinearRegression(new Data(data));
                double[][] clearData = new double[data.Length - countErrors(data)][];
                double[][] posPredInt = regr.getPosPredInt();
                double[][] negPredInt = regr.getNegPredInt();
                int j = 0;
                for (int i = 0; i < data.Length; i++)
                {
                        if (data[i][1] > posPredInt[i][1] || data[i][1] < negPredInt[i][1])
                        continue;
                        clearData[j] = new double[2];
                        clearData[j][0] = data[i][0];
                        clearData[j][1] = data[i][1];
                        j++;
                }
                return clearData;
        }
    }
}
```

Intervals.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 Intervals
    {
        public Intervals(RegressionModel regression, Chart chart)
```

```
            {
                int indX = 0;
                int indY = 1;
                double[][] graph = regression.getRegression();
                double[][] posConfInt = regression.getPosConfInt();
                double[][] negConfInt = regression.getNegConfInt();
                double[][] posPredInt = regression.getPosPredInt();
                double[][] negPredInt = regression.getNegPredInt();
                int N = graph.Length;

                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(posConfInt[i][indX], posConfInt[i][indY]);
                    chart.Series[2].Points.AddXY(negConfInt[i][indX], negConfInt[i][indY]);
                    chart.Series[3].Points.AddXY(posPredInt[i][indX], posPredInt[i][indY]);
                    chart.Series[4].Points.AddXY(negPredInt[i][indX], negPredInt[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(posPredInt[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 getConfIntDelta(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.getAvgX();
		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 getPredIntDelta(double x)
{
		return t * getS() * Math.Sqrt(1 + 1 / data.getN() + Math.Pow(x - data.getAvgX(), 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];
			regression[i][0] = data.getX(i);
			regression[i][1] = getYCalculated(regression[i][0]);
		}
		return regression;
}

public override double[][] getPosConfInt()
{
		double[][] positiveConfInt = new double[data.getN()][];
		for (int i = 0; i < data.getN(); i++)
		{
			positiveConfInt[i] = new double[2];
			double x = data.getX(i);
			positiveConfInt[i][0] = x;
			positiveConfInt[i][1] = getYCalculated(x) + getConfIntDelta(x);
		}
		return positiveConfInt;
}

public override double[][] getNegConfInt()
{
		double[][] negativeConfInt = new double[data.getN()][];
		for (int i = 0; i < data.getN(); i++)
		{
```

```
                negativeConfInt[i] = new double[2];
                double x = data.getX(i);
                negativeConfInt[i][0] = x;
                negativeConfInt[i][1] = getYCalculated(x) - getConfIntDelta(x);
            }
            return negativeConfInt;
        }

        public override double[][] getPosPredInt()
        {
            double[][] positivePredInt = new double[data.getN()][];
            for (int i = 0; i < data.getN(); i++)
            {
                positivePredInt[i] = new double[2];
                double x = data.getX(i);
                positivePredInt[i][0] = x;
                positivePredInt[i][1] = getYCalculated(x) + getPredIntDelta(x);
            }
            return positivePredInt;
        }

        public override double[][] getNegPredInt()
        {
            double[][] negativePredInt = new double[data.getN()][];
            for (int i = 0; i < data.getN(); i++)
            {
                negativePredInt[i] = new double[2];
                double x = data.getX(i);
                negativePredInt[i][0] = x;
                negativePredInt[i][1] = getYCalculated(x) - getPredIntDelta(x);
            }
            return negativePredInt;
        }
    }
}
```

NonlinearRegression.cs

```
using System;
namespace Regression
{
        class NonlinearRegression : RegressionModel
        {
            private LinearRegression linearRegression;
            public NonlinearRegression(LinearRegression _linearRegression) : base(new
        Data(LgNormalizer.reverseData(_linearRegression.getData().getData())))
            {
                linearRegression = _linearRegression;
            }
```

```
public override double[][] getNegConfInt()
{
	return lgNormalizer.reverseData(linearRegression.getNegConfInt());
}

public override double[][] getNegPredInt()
{
	return lgNormalizer.reverseData(linearRegression.getNegPredInt());
}

public override double[][] getPosConfInt()
{
	return lgNormalizer.reverseData(linearRegression.getPosConfInt());
}

public override double[][] getPosPredInt()
{
	return lgNormalizer.reverseData(linearRegression.getPosPredInt());
}

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

```
		{
			double lgNormalizedX = lgNormalizer.lgNormalize(x);
			return lgNormalizer.reverse(linearRegression.getYCalculated(lgNormalizedX) +
		linearRegression.getPredIntDelta(lgNormalizedX)) - getYCalculated(x);
		}

		public double getConfIntDelta(double x)
		{
			 double lgNormalizedX = lgNormalizer.lgNormalize(x);
			 return lgNormalizer.reverse(linearRegression.getYCalculated(lgNormalizedX) +
	linearRegression.getConfIntDelta(lgNormalizedX)) - getYCalculated(x);
		}
	}
}
```

Додаток В – Опис програми

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування програми - «Нелінійна регресійна модель для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд», що розробляється на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

Позначення програми - «Nonlinear Regression».

1.2 Мови програмування

Програмний додаток для оцінювання розміру персональних застосунків-органайзерів, що ствюються мовою Kotlin для Андроїд, було розроблено об’єктно-орієнтовною мовою програмування C#. Для розробки програмного забезпечення обрали інтерфейс програмування додатків (API) Windows Forms, що є частиною .NET Framework та сердовище розробки Visual Studio.

2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація процесу оцінювання розміру персональних застосунків-органайзерів. Програмний додаток виконує наступні перелічені функції:

– надання можливість вносити дані про проекти персональних застосунків-органайзерів (кількість рядків коду та кількість класів);

– виконання нормалізації внесених даних за допомогою нормалізуючого перетворення на основі десяткового логарифму;

– видалення викидів за наявності;

– побудова лінійної регресійної моделі;

– побудова довірчих інтервалів на основі нормалізованих даних;

– побудова інтервалів передбачення на основі нормалізованих даних;

– побудова нелінійної регресійної моделі на основі лінійного рівняння регресії;

– розрахування метрик якості;

– оцінювання розміру персональних застосунків-органайзерів;

– виведення отриманих розрахунків на екран користувача.

3 Технічні засоби, що використовуються

Програмний додаток «Nonlinear Regression» потребує від комп'ютеру, на якому він буде встановлений, наступних характеристик, які треба розглядати як мінімальні:

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 500 MB вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7. Необхідна наявність .NET Framework версії 4.6 або вище. Швидкість та якість роботи програмного додатку на конкретному комп'ютері залежить також від характеристик окремих його комплектуючих.

4 Виклик та завантаження

Виклик та завантаження програмного додатку «Nonlinear Regression» відбувається запуском відповідно названого виконуваного файлу. Запуск програмного додатку «Nonlinear Regression» іншими способами неможливий.

5 Вхідні дані

Вхідні дані: введення даних здійснюється за допомогою пристроїв введення даних (клавіатура та миша). Вхідні дані для програми мають вводитись у вигляді файлу, який містить наступні стовбці значень: кількість рядків коду, кількість класів. Також користувач має вводити значення кількості

класів для обчислення оцінки розміру персонального застосунку-органайзера, що створюється мовою Kotlin для Андроїд.

6 Вихідні дані

Вихідні дані: виведення даних здійснюється за допомогою пристрою виведення даних (монітор). Вихідними даними є результат обчислень у вигляді графіків та числових значень, що відображаються на екрані користувача.

Додаток Г - Інструкція користувача

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

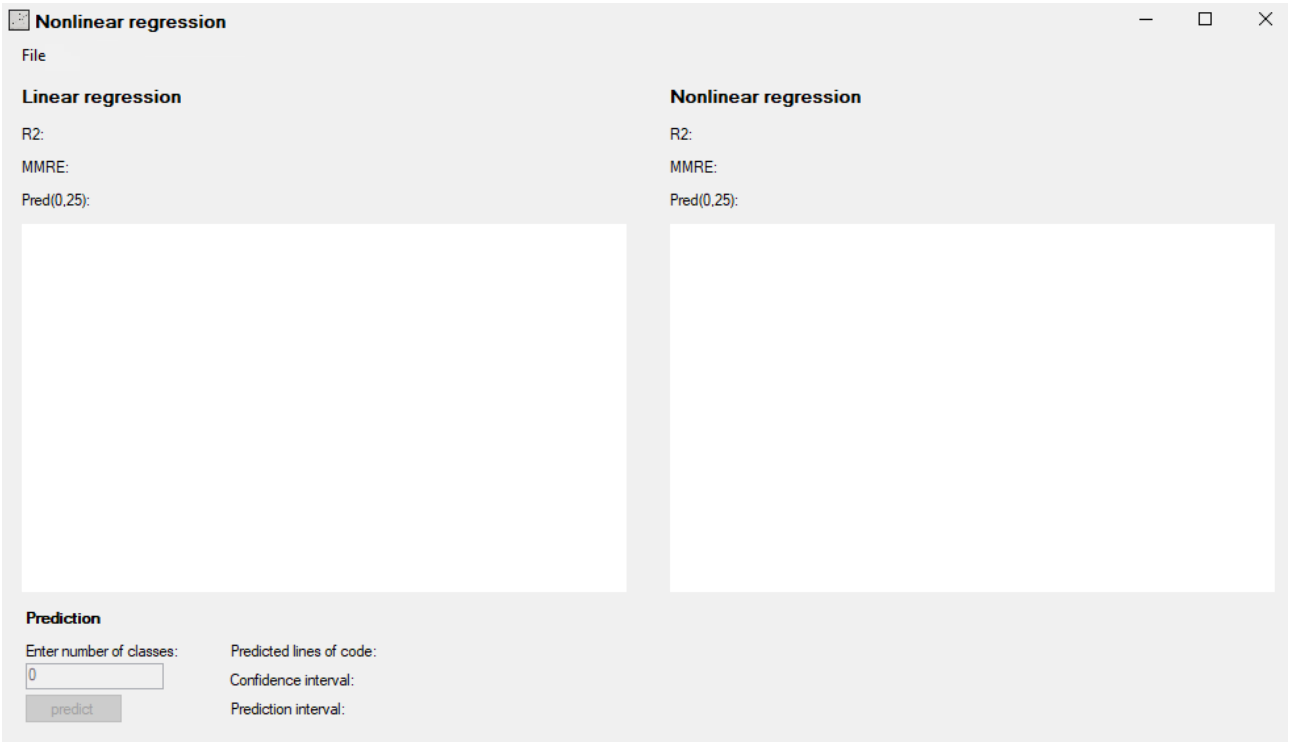

Рисунок Г.1 – Головний екран інтерфейсу користувача

Користувач має можливість завантажити файл з вхідними даними. Після завантаження файлу з вхідними даними програма надає можливість здійснити нормалізацію вхідних даних, видалити викиди за наявності, побудувати лінійне рівняння регресії, побудувати довірчі інтервали та інтервали передбачення, побудувати нелінійну регресію та відповідні інтервали, розрахувати показники оцінки якості регресії, а також здійснити оцінювання розміру персональних застосунків-органайзерів.

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 та коректною сруктурою даних файлу.

У випадку завантаження файлу з некоректним типом розширенням або файлу з неправильною структурою даних, користувач отримає повідомлення про помилку, зображене на рис. Г.2.

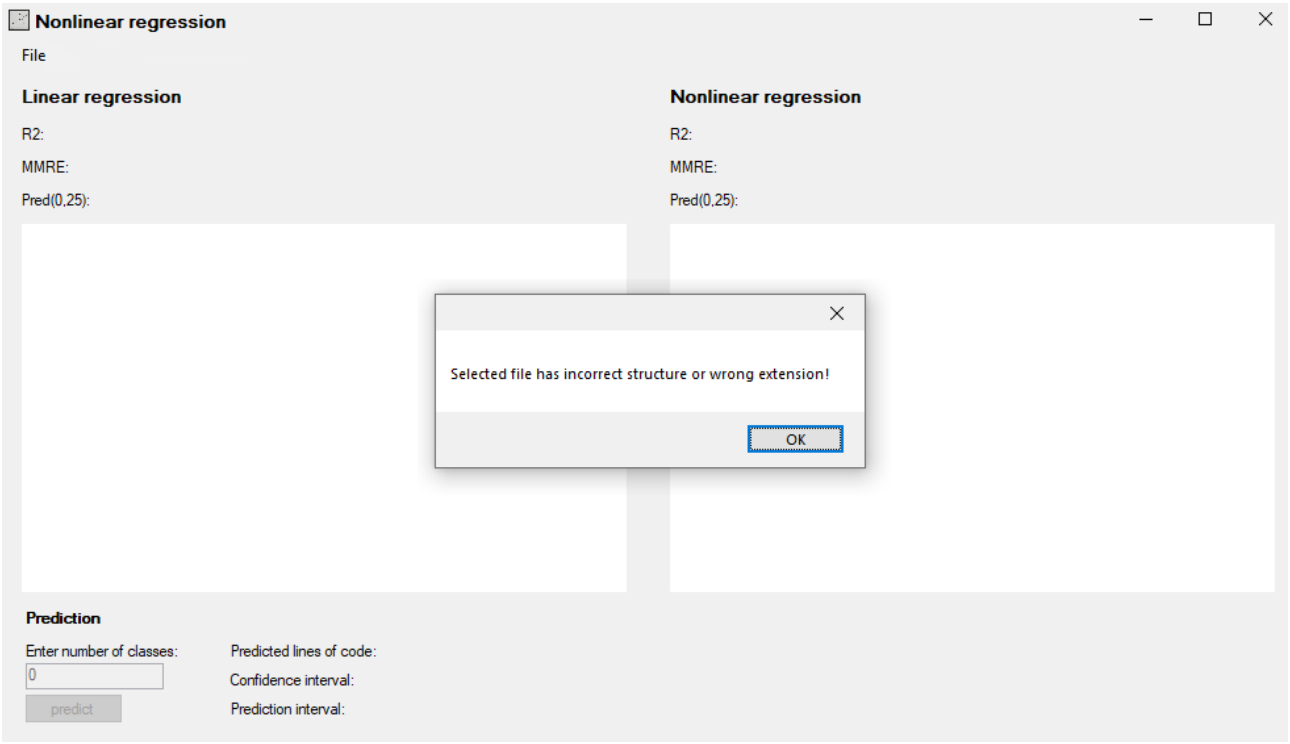

Рисунок Г.2 – Повідомлення про помилку

Якщо вхідні дані завантажено успішно, система нормалізує вхідні дані за допомогою нормалізуючого перетворення на основі десяткового логарифму та видаляє 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). Після побудови нелінійної регресійної моделі функція оцінки розміру персонального застосунку-органайзера стане активною та доступною для використання. Виведені на екран результати розрахунків після завантаження вхідних даних зображені на рис. Г.3.

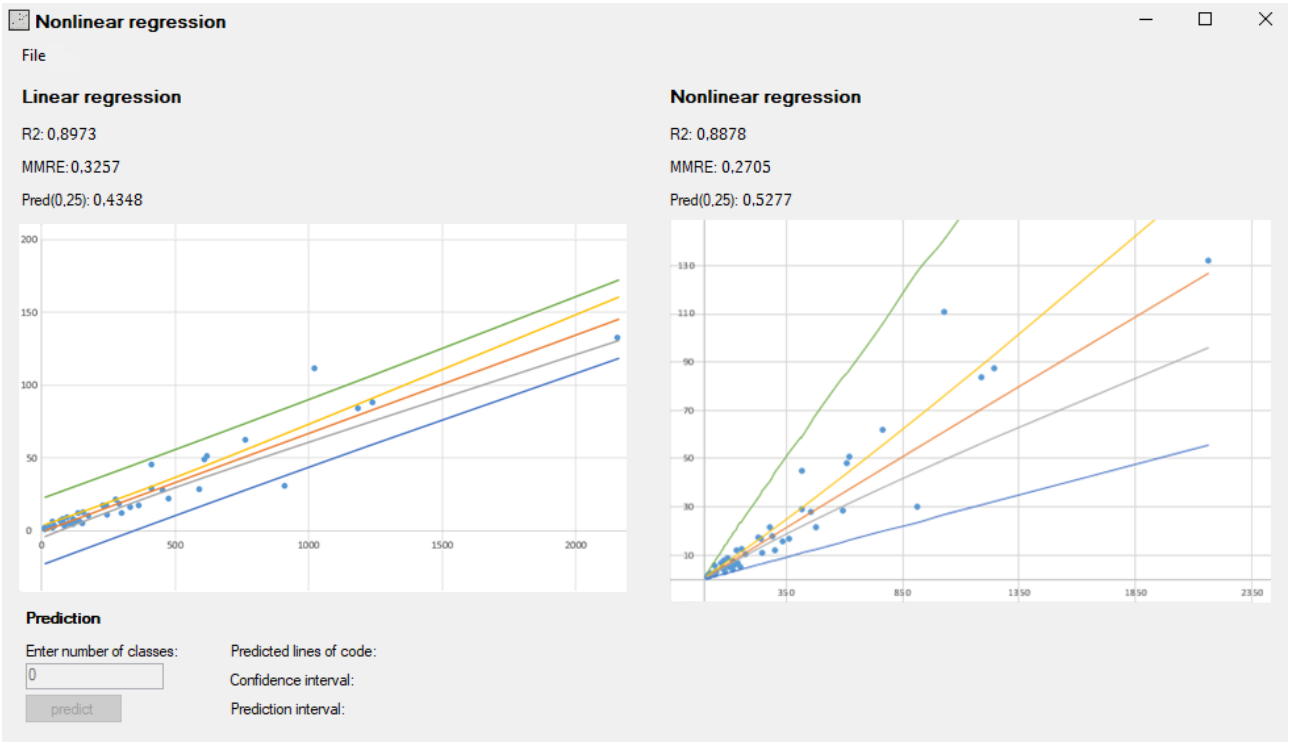

Рисунок Г.3 – Результати розрахунків

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

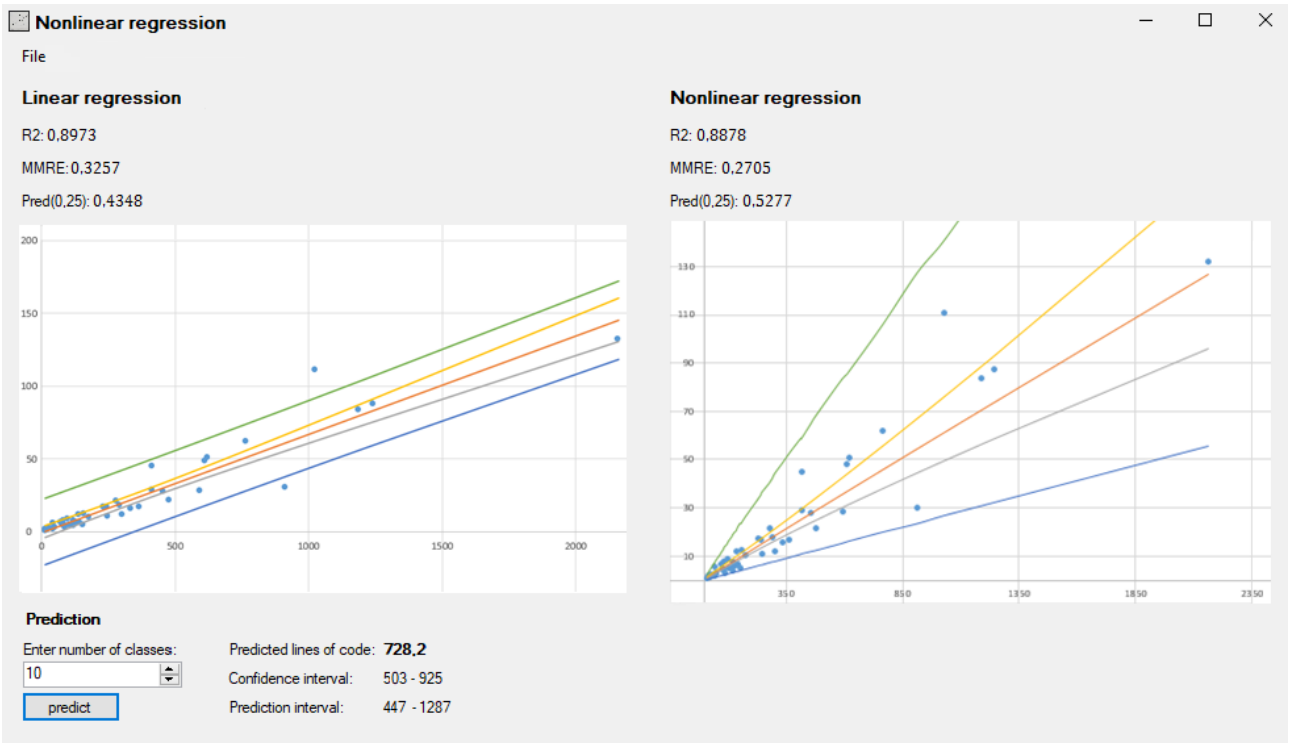

Рисунок Г.4 – Результат оцінювання розміру персонального застосунку-органайзера

У результаті розрахунку оцінки розміру персонального застосунку-органайзера на екран користувача буде виведено 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 програмного забезпечення

1 Об’єкт випробування

Об’єктом випробування є програма для оцінювання розміру персональних застосунків-органайзерів що створюються мовою Kotlin для Андроїд, що розробляється на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного забезпечення, перевірка відповідності характеристик та вимог розробленої програми, викладених у технічному завдані, приведення прикладу роботи та отримання відповідних навичок.

3 Вимоги до додатка

При проведенні випробувань функціональні характеристики (можливості) програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик»технічного завдання.

4 Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

1) Технічне завдання.

2) Текст програми.

3) Опис програми.

4) Програма та методика випробувань.

5) Інструкція користувача.

5 Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 500 MB вільного місця.

5.2 Програмні засоби, що використовуються під час випробувань

Склад програмних засобів, що використовувалися під час випробувань: операційна система - ОС Windows версії не нижче 7.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» Додатку А - Технічне завдання. Перевірка вважається завершеною у разі відповідності складу і послідовності виконаних дій пункту «Вимоги до функціональних характеристик». В процесі тестування була перевірена функціональність за темою програмного забезпечення для оцінювання розміру персональних застосунків-органайзерів, що створюються мовою Kotlin для Андроїд. У табл. Д. 1, що наведена нижче, вказано перелік випробувань основних функціональних можливостей.

Таблиця Д.1-Перелік випробувань основних функціональних можливостей

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